ALGORYTMY

VOL. VII • NO. 13 • 1970

INSTYTUT MASZYN MATEMATYCZNYCH

A L G O R Y T M Y

Vol. VII N° 13 1970

Adres Redakcji: Warszawa, ul. Krzywickiego 34, tel. 28-37-29

T R E Ś Ć

С О Д Е Р Ж А Н И Е

C O N T E N T S

ALGORYTMY, VII, N° 13 1970 , 5-7

A MONTE CARLO ESTIMATION OF THE MAXIMUM OF A FUNCTION

by Ryszard ZIELIŃSKI

Received October 23rd, 1970

Evaluation of the maximum of a function is formulated as a problem of estimation of the mode of a probability distribution.

The Monte Carlo method is usually defined as a replacement of the given numerical problem by a statistical problem of estimation of a parameter of a hypothetical population. The latter problem is solved using a sample taken from the population. The well known algorithms for solving optimization problems (see e.g. [5]) are rather random search procedures than algorithms of a parameter estimation. In what follows the problem of evaluation of the maximum of a function will be formulated as a problem of estimation of the mode of a density function of a random variable.

Let D be a bounded domain in m-dimensional Euclidean space and let F be a bounded real-valued function defined on D. The point at which F reaches maximum is to be evaluated.

Let (a, b), $a = (a_1, \ldots, a_m)$, $b = (b_1, \ldots, b_m)$, be a m-dimensional interval $\{x=(x_1, \ldots, x_m): a_j < x_j < b_j,\ j = 1, 2, \ldots, m\}$ containing the domain D and let c be a constant such that $F(x) < c$, $x \in D$. Define $D^+ = \{x \in D: F(x) > 0\}$ and suppose $\mathrm{vol}(D^+) > 0$; if it is not the case, we can consider $F(x) + \alpha$

instead of $F(x)$ where α is a suitable constant. Define $g(x) = F(x)$ for $x \in D^+$, $= 0$ for $x \in (a, b) - D^+$ and suppose $\lambda = \int g(x)dx$ exists. Let $\xi = (\xi_0, \xi_1, \dots, \xi_m)$ be a random variable distributed uniformly on the $(m+1)$-dimensional interval $(0,c) \times (a,b) = \{(x_0, x_1, \dots, x_m): 0 < x_0 < c, a_j < x_j < b_j, j = 1, 2, \dots, m\}$ and define an m-dimensional random variable $\zeta = (\zeta_1, \dots, \zeta_m)$ as follows:

$$\mathrm{Prob}\{\zeta_1 < y_1, \dots, \zeta_m < y_m\} =$$

$$= \mathrm{Prob}\{\xi_1 < y_1, \dots, \xi_m < y_m \mid \xi_0 < F(\xi_1, \dots, \xi_m)\}.$$

It is easy to see that the random variable ζ is distributed on the interval (a, b) with the density g/λ. Now the problem of evaluating the maximum of F is equivalent to the problem of estimation of the mode of the distribution with the density g/λ. This can be solved by one of the known methods (e.g. [2], [3], [4]). Sampling from the distribution of random variable ζ is very easy: points $(x_0, x_1, \dots, x_m)$ from the interval $(0, c) \times (a, b)$ are to be sampled according to the uniform distribution; points satisfying $x_0 < F(x_1, \dots, x_m)$ are accepted and $(x_1, \dots, x_m)$ is taken as the observed value of the random variable ζ; other points are rejected [1].

References

[1] BUTLER J.W.: Machine Sampling from Given Probability Distribution. Symposium on Monte Carlo Methods. H.A. Meyer /ed./. New York-London, Wiley, Chapman a. Hall, 1956.

[2] MOORE D.S., HENRICHON E.G.: Uniform Consistency of Some Estimates of a Density Function. Ann. Math. Statist. 40 1969 1499-1502.

[3] VAN RYZIN J.: On Strong Consistency of Density Estimates. Ann. Math. Statist. 40 1969 1765-1772.

[4] VENTER H.J.: On Estimation of the Mode. Ann. Math. Statist. 38 1967 1446-1455.

[5] ZIELIŃSKI R.: Metody Monte Carlo. Warszawa, WNT 1970.

PEWNA METODA MONTE CARLO SZACOWANIA MAKSIMUM FUNKCJI

Streszczenie

Obliczanie maksimum funkcji sformułowano jako zagadnienie szacowania wartości modalnej pewnego rozkładu prawdopodobieństwa.

МЕТОД МОНТЕ КАРЛО ДЛЯ ОЦЕНКИ МАКСИМУМА ФУНКЦИИ

Резюме

Вычисление максимума функции сформулировано как вопрос оценки моды некоторого распределения вероятности.

R.Z.

ALGORYTMY, VII, N° 13 1970, 9-16

DYSKRETNE PROCESY WSPÓŁDZIAŁANIA

Józef WINKOWSKI

Pracę złożono 70.11.4

W pracy definiuje się pewną relację między maszynami i rodzinami maszyn i wykorzystuje się ją do opisywania procesów współdziałania.

1. WSTĘP

Będziemy zajmować się zagadnieniem opisywania procesów dość typowych dla symulacji (por. [1]). Najogólniej biorąc, chodzi o procesy współdziałania obiektów wzajemnie planujących swoje akcje. Każdy obiekt jest tego rodzaju, że każda jego akcja odbywa się w pewnej chwili i prócz zaplanowania akcji innych obiektów, polega na przetworzeniu informacji dostępnej obiektowi. Zmiany informacji są wynikiem akcji obiektów i mają charakter dyskretny. Z tego powodu omawiane procesy są nazywane dyskretnymi procesami współdziałania. Rozważamy obiekty stanowiące maszyny w sensie [3], a procesy ich współdziałania opisujemy konstruując maszyny będące wiązkami obiektów.

2. POJĘCIA OGÓLNE

W dalszym ciągu $\emptyset$ będzie oznaczać zbiór pusty, xf wartość funkcji f dla x a Dom_f dziedzinę funkcji f. Równość $\tau_1 | = | \tau_2$ będzie oznaczać, że wyrażenie τ_1 posiada określoną wartość wtedy i tylko wtedy, gdy wyrażenie τ_2 posiada określoną wartość i że wartości te są identyczne.

<u>Struktura</u> będziemy nazywać zbiór wraz z określonymi na nim relacjami, operacjami i funkcjami (być może wieloargumentowy-

i częściowymi), przy czym sam ten zbiór będziemy nazywać nośnikiem struktury.

Relacją porządku liniowego na zbiorze X będziemy nazywać zwrotną przechodnią relację dwuargumentową $\leqslant$ na X taką, że $x \leqslant y$ lub $y \leqslant x$ a poza tym $x \leqslant y$, $y \leqslant x$ pociąga $x = y$ dla każdych dwóch elementów zbioru X. Jeśli w każdym niepustym $Y \subset X$ istnieje element najmniejszy, to będziemy mówić, że relacja taka dobrze porządkuje zbiór X.

Operacją następstwa w dobrze uporządkowanym przez relację $\leqslant$ zbiorze X będziemy nazywać funkcję f określoną dla takich $a \in X$, dla których istnieje $x \in X$, $x \geqslant a$, $x \neq a$ i taką, że $f(a)$ jest najmniejszym spośród elementów $x \in X$, $x \geqslant a$, $x \neq a$. Tak np. $f = \{\langle x_1, x_2\rangle, \dots, \langle x_{n-1}, x_n\rangle\}$ jest operacją następstwa w zbiorze o elementach $x_1 \leqslant x_2 \leqslant \dots \leqslant x_{n-1} \leqslant x_n$.

Systemem wyboru będziemy nazywać każdą strukturę $\langle X, f\rangle$ o nośniku X i częściowej operacji jednoargumentowej f, jeśli f jest operacją następstwa w X dobrze uporządkowanym przez pewną relację $\leqslant$. Będziemy mówić, że system $\langle X, f\rangle$ wybiera element x, jeśli $X \neq \emptyset$ i x jest najmniejszym elementem zbioru X. Tak np. $\langle \{x_1, x_2, \dots x_{n-1}, x_n\}, f\rangle$ z $f = \{\langle x_n, x_{n-1}\rangle, \dots \langle x_2, x_1\rangle\}$ jest systemem wyboru wybierającym x_n.

Maszyną będziemy nazywać każdą strukturę $\langle \Sigma, \varphi\rangle$ o nośniku Σ i częściowej operacji jednoargumentowej φ (por. [3]). Elementy zbioru Σ będziemy nazywać stanami albo inaczej sytuacjami maszyny.

Będziemy mówić, że maszyna $\langle \Sigma, \varphi\rangle$ jest wiązką rodziny maszyn $\langle \Sigma_x, \varphi_x\rangle$ $(x \in X)$, jeśli istnieje funkcja $\mathcal{W}$ przyporządkowująca każdemu $\sigma \in \Sigma$ system wyboru $\langle Q_\sigma, \mathcal{N}_\sigma\rangle$ z $Q_\sigma \subset X$, dla $x \in X$ istnieją funkcje $pr_x \subset \Sigma \times \Sigma_x$ oraz dla każdego $\sigma \in \Sigma, \sigma' \in \Sigma$

(W_1) $\sigma pr_x = \sigma' pr_x$ dla $x \in X$, $Q_\sigma = Q_{\sigma'}$, $\mathcal{N}_\sigma = \mathcal{N}_{\sigma'}$, pociąga $\sigma = \sigma'$

(W_2) $\sigma\varphi$ jest określone wtedy i tylko wtedy gdy system $\langle Q_\sigma, \mathcal{N}_\sigma \rangle$ wybiera pewien element x i $\sigma pr_x \varphi_x$ jest określone; wówczas $\sigma\varphi pr_x = \sigma pr_x \varphi_x$.

Funkcję $\mathcal{W}$ będziemy nazywać funkcją wyboru, a funkcje pr_x projekcjami.

3. WIĄZKI OPISUJĄCE PROCESY

Zajmiemy się teraz konstruowaniem maszyn będących wiązkami rodzin maszyn i opisujących pewne procesy współdziałania.

Niech $\Omega_1 = \{\alpha, \dots\}, \Omega_2 = \{\beta, \dots\} \ni \mathcal{T}$ i niech $\mathcal{M} = \langle A; X, X_\beta, \dots, \leqslant \rangle$ będzie strukturą z rozbiciem $\{X, X_\beta, \dots\}$ zbioru A i z relacją $\leqslant$ porządku liniowego na $X_\mathcal{T}$. Elementy zbioru $X_\mathcal{T}$ będziemy nazywać chwilami.

Niech $\langle \Sigma, \varphi \rangle$ będzie maszyną taką, że

(X_1) dla częściowych funkcji $\mathcal{N}, \alpha, \dots$ z $\Sigma \times X$ do X i częściowych funkcji $\beta, \dots$ z $\Sigma \times X$ do $X_\beta, \dots$, niech $\mathcal{N}(\sigma, \cdot) = \mathcal{N}(\sigma', \cdot)$, $\alpha(\sigma, \cdot) = \alpha(\sigma', \cdot), \dots, \beta(\sigma, \cdot) = \beta(\sigma', \cdot), \dots$ pociąga $\sigma = \sigma'$,

(X_2) dla każdego $\sigma \in \Sigma$ struktura $\langle Dom_{\mathcal{T}(\sigma, \cdot)}, \mathcal{N}(\sigma, \cdot) \rangle$ jest systemem wyboru, $\mathcal{T}(\sigma, x) \leqslant \mathcal{T}(\sigma, \mathcal{N}(\sigma, x))$ jeśli tylko oba wyrażenia są określone i $\mathcal{T}(\sigma, x_0) \leqslant \mathcal{T}(\sigma\varphi, x)$ jeśli tylko oba wyrażenia są określone, i jeśli system $\langle Dom_{\mathcal{T}(\sigma, \cdot)}, \mathcal{N}(\sigma, \cdot) \rangle$ wybiera x_0.

Wówczas każdą sytuację σ tej maszyny można utożsamiać ze strukturą $\langle N_\sigma ; \mathcal{N}_\sigma, \alpha_\sigma, \dots ; \beta_\sigma, \dots \rangle$ o nośniku N_σ złożonym z elementów zbioru X występujących w dziedzinach lub w zbiorach wartości funkcji $\mathcal{N}(\sigma, \cdot), \alpha(\sigma, \cdot), \dots, \beta(\sigma, \cdot), \dots$, o częściowych operacjach jednoargumentowych $\mathcal{N}_\sigma = \mathcal{N}(\sigma, \cdot), \alpha_\sigma = \alpha(\sigma, \cdot), \dots$ i częściowych funkcjach $\beta_\sigma = \beta(\sigma, \cdot), \dots$.

Dla każdego $x \in X$ niech $\langle \Sigma_x, \varphi_x \rangle$ będzie maszyną taką, że:

(x_1) dla częściowych funkcji $\alpha_x, \ldots$ z $\Sigma_x \times X$ do X i częściowych funkcji $\beta_x, \ldots$ z $\Sigma_x \times X$ do $X_\beta, \ldots$, jeśli $\alpha_x(\sigma, \cdot) = \alpha_x(\sigma', \cdot), \ldots, \beta_x(\sigma, \cdot) = \beta_x(\sigma', \cdot), \ldots$, to $\sigma = \sigma'$,

(x_2) jeśli φ_x jest określona dla σ, to $\mathcal{T}_x(\sigma, x)$ jest określone i zbiór N_σ złożony z elementów zbioru X występujących w dziedzinach lub w zbiorach wartości funkcji $\alpha_x(\sigma, \cdot), \ldots, \beta_x(\sigma, \cdot) \ldots$ jest generowany przez x za pomocą operacji $\alpha_x(\sigma, \cdot), \ldots$.

Wówczas każdą sytuację σ maszyny $\langle \Sigma_x, \varphi_x \rangle$ można utożsamiać ze strukturą $\langle N_\sigma; \alpha_\sigma, \ldots, \beta_\sigma, \ldots \rangle$ o nośniku N_σ, częściowych operacjach jednoargumentowych $\alpha_\sigma = \alpha_x(\sigma, \cdot), \ldots$ i częściowych funkcjach $\beta_\sigma = \beta_x(\sigma, \cdot), \ldots$. Element x będziemy nazywać <u>reprezentantem</u> maszyny $\langle \Sigma_x, \varphi_x \rangle$.

Dla każdego $x \in X$ niech $pr_x \subset \Sigma \times \Sigma_x$ będzie funkcją (być może częściową) przyporządkowującą sytuacji $\sigma = \langle N_\sigma; \mathcal{N}_\sigma, \alpha_\sigma, \ldots; \beta_\sigma, \ldots \rangle$ maszyny $\langle \Sigma, \varphi \rangle$ sytuację $\sigma pr_x = \langle N_{\sigma pr_x}; \alpha_{\sigma pr_x}, \ldots; \beta_{\sigma pr_x}, \ldots \rangle$ maszyny $\langle \Sigma_x, \varphi_x \rangle$ taką, że $N_{\sigma pr_x} \subset N_\sigma, \alpha_{\sigma pr_x} \subset \alpha_\sigma, \ldots, \beta_{\sigma pr_x} \subset \beta_\sigma, \ldots$ (istnienie takiej funkcji zależy od funkcji $\alpha, \ldots, \beta, \ldots$ i od funkcji $\alpha_x, \ldots, \beta_x, \ldots$). Jeśli dla każdej sytuacji $\sigma \in \Sigma$ spełnione są warunki:

(1) $\bigcup_x N_{\sigma pr_x} = N_\sigma$, $\bigcup_x \alpha_{\sigma pr_x} = \alpha_\sigma, \ldots, \bigcup_x \beta_{\sigma pr_x} = \beta_\sigma, \ldots$ z sumowaniami rozciągniętymi na te $x \in X$, dla których σpr_x jest określone,

(2) $\sigma\varphi$ jest określone wtedy i tylko wtedy, gdy system wyboru $\langle Dom_{\mathcal{T}_\sigma}, \mathcal{N}_\sigma \rangle$ wybiera pewien element $x \in X$ i $\sigma pr_x \varphi_x$ jest określone; wówczas $N_{\sigma\varphi} = (N_\sigma \setminus N_{\sigma pr_x}) \cup N_{\sigma pr_x \varphi_x}$, $\alpha_{\sigma\varphi} = (\alpha_\sigma \setminus \alpha_{\sigma pr_x}) \cup \alpha_{\sigma pr_x \varphi_x}, \ldots, \beta_{\sigma\varphi} = (\beta_\sigma \setminus \beta_{\sigma pr_x}) \cup \beta_{\sigma pr_x \varphi_x}, \ldots$ oraz $N_{\sigma\varphi pr_x} = N_{\sigma\varphi} \setminus (N_\sigma \setminus N_{\sigma pr_x})$, $\alpha_{\sigma\varphi pr_x} = \alpha_{\sigma\varphi} \setminus (\alpha_\sigma \setminus \alpha_{\sigma pr_x}), \ldots$ $\beta_{\sigma\varphi pr_x} = \beta_{\sigma\varphi} \setminus (\beta_\sigma \setminus \beta_{\sigma pr_x}), \ldots$,

to $\langle \Sigma, \varphi \rangle$ jest wiązką rodziny maszyn $\langle \Sigma_x, \varphi_x \rangle$ $(x \in X)$ z funkcją wyboru przyporządkowującą każdej sytuacji $\sigma \in \Sigma$ system wyboru $\langle Dom_{\mathcal{T}_\sigma}, \mathcal{N}_\sigma \rangle$ oraz z projekcjami pr_x . Miano-

wicie, jeśli $\sigma\varphi$ jest określone, to dla x wybieranego przez $\langle Dom_{\mathcal{T}_\sigma}, \mathcal{N}_\sigma \rangle$ mamy $N_{\sigma\varphi pr_x} = N_{\sigma pr_x \varphi_x}$, $\alpha_{\sigma\varphi pr_x} = \alpha_{\sigma pr_x \varphi_x}, \dots,$ $\beta_{\sigma\varphi pr_x} = \beta_{\sigma pr_x \varphi_x}, \dots$. Ponadto, wobec (1), spełniony jest warunek (W1).

Opisana wyżej maszyna $\langle \Sigma, \varphi \rangle$ posiada interpretację dynamiczną, dzięki której może być uważana za opis dyskretnego procesu współdziałania maszyn $\langle \Sigma_x, \varphi_x \rangle (x \in X)$. Niech mianowicie $\sigma_0 \in \Sigma$ będzie pewną sytuacją. Jeśli operacja φ jest określona dla σ_0 , to system wybierania $\langle Dom_{\mathcal{T}_{\sigma_0}}, \mathcal{N}_{\sigma_0} \rangle$ wybiera $x_0 \in N\sigma_0$ dające minimum funkcji $\mathcal{T}_{\sigma_0}$ i operacja φ przekształca część $\sigma_0 pr_{x_0} \subset \sigma_0$ stanowiącą sytuację maszyny $\langle \Sigma_{x_0}, \varphi_{x_0} \rangle$ tak jak operacja φ_{x_0} . Wynikiem działania φ jest nowa sytuacja $\sigma_1 = \sigma_0 \varphi$, w której być może jest wybierany element $x_1 \in N\sigma_1$ itd. Wobec (X_2) maszyna $\langle \Sigma, \varphi \rangle$ ma tę własność, że dla każdej sytuacji $\sigma_i \in \Sigma$ jest $x_i \mathcal{T}_{\sigma_i} \leqslant y \mathcal{T}_{\sigma_i}$ dla wszystkich $y \in N_{\sigma_i}$ oraz $x_0 \mathcal{T}_{\sigma_0} \leqslant x_1 \mathcal{T}_{\sigma_1} \leqslant \dots$ Zatem sytuacji σ_0 można przyporządkować następującą funkcję częściową ρ_{σ_0} na X_β:

$$\rho_{\sigma_0}(t) = \sigma_{i_k} \quad \text{dla} \quad x_{i_k} \mathcal{T}_{\sigma i_k} < t < x_{i_{k+1}} \mathcal{T}_{\sigma i_{k+1}}, \quad k = 0, 1, \dots,$$

gdzie

$$x_0 \mathcal{T}_{\sigma 0} = \dots = x_{i_0} \mathcal{T}_{\sigma i_0} < x_{i_0+1} \mathcal{T}_{\sigma i_0+1} = \dots = x_{i_1} \mathcal{T}_{\sigma i_1} < x_{i_1+1} \mathcal{T}_{\sigma i_{1+1}} = \dots,$$

zwaną <u>przebiegiem</u> maszyny $\langle \Sigma, \varphi \rangle$ o <u>sytuacji początkowej</u> σ_0 . Dziedziną przebiegu jest suma przylegających przedziałów otwartych $(t_0, t_1), (t_1, t_2), \dots$, a na każdym z nich wartość przebiegu jest stała i jest pewną sytuacją maszyny $\langle \Sigma, \varphi \rangle$. W chwilach $t_0, t_1, \dots$, które można interpretować jako chwile akcji, wartość przebiegu nie jest określona. Przebieg maszyny $\langle \Sigma, \varphi \rangle$ o sytuacji początkowej $\sigma_0 \in \Sigma$ jest w istocie przebiegiem dyskretnego procesu współdziałania i maszyna $\langle \Sigma, \varphi \rangle$ stanowi opis tego procesu. Wartości funkcji $\mathcal{T}_\sigma$ można interpretować jako chwile planowanych akcji maszyn w sytuacji σ.

4. PRZYKŁAD

Niech $\xi = \langle U, S, V, \lambda_\xi, \mu_\xi \rangle$, $\eta = \langle U, S, V, \lambda_\eta, \mu_\eta \rangle$ będą automatami o alfabecie wejściowym U , zbiorze stanów S , alfabecie wyjściowym $V = U$, funkcjach przejścia $(u, s) \to \lambda_\xi(u, s)$,

$(u,s)\to\lambda_\eta(u,s)$ i funkcjach wyjścia $(u,s)\to\mu_\xi(u,s)$, $(u,s)\to\mu_\eta(u,s)$. Niech każdy z nich pod wpływem sygnału na wejściu przechodzi po pewnym czasie reakcji do nowego stanu, wytwarzając równocześnie sygnał na wyjściu. Ponadto niech każdy z automatów pamięta ostatni z sygnałów otrzymanych na wejściu. Niech czasy reakcji automatów będą odpowiednią wartościami funkcji $(u,s)\to\tau_\xi(u,s)$, $(u,s)\to\tau_\eta(u,s)$.

Rozważmy proces współdziałania obu automatów, gdy każdy z nich przekazuje sygnał wytworzony na wyjściu na wejście drugiego.

Niech $X=\{x,y\}$ i niech x reprezentuje automat ξ a element y automat η. Niech Σ będzie zbiorem wszystkich struktur $\sigma=\langle N_\sigma;\mathcal{N}_\sigma,\alpha_\sigma;\beta_\sigma,\gamma_\sigma,\mathcal{T}_\sigma\rangle$, gdzie $N_\sigma=\{x,y\}$, $\alpha_\sigma=\{\langle x,y\rangle,\langle y,x\rangle\}$, $\beta_\sigma=\{\langle x,u_x\rangle,\langle y,u_y\rangle\}$, $\gamma_\sigma=\{\langle x,s_x\rangle,\langle y,s_y\rangle\}$, $\mathcal{T}_\sigma=\{\langle x,t_x\rangle,\langle y,t_y\rangle\}$, $u_x\in U$, $u_y\in U$, $s_x\in S$, $s_y\in S$ i t_x,t_y są rzeczywiste i gdzie $\langle\{x,y\},\mathcal{N}_\sigma\rangle$ jest systemem wyboru wybierającym element dający minimum $\mathcal{T}_\sigma$. Niech φ będzie funkcją przyporządkowującą każdej takiej strukturze strukturę $\sigma\varphi$, gdzie $N_{\sigma\varphi}=N_\sigma$, $\alpha_{\sigma\varphi}=\alpha_\sigma$ oraz $\beta_{\sigma\varphi}=\{\langle x,u_x\rangle,\langle y,\mu_\xi(u_x,s_x)\rangle\}$, $\gamma_{\sigma\varphi}=\{\langle x,\lambda_\xi(u_x,s_x)\rangle,\langle y,s_y\rangle\}$, $\mathcal{T}_{\sigma\varphi}=\{\langle x,t_x+\tau_\xi(u_x,s_x)\rangle,\langle y,t_y\rangle\}$, gdy wybierany jest element x i $\beta_{\sigma\varphi}=\{\langle x,\mu_\eta(u_y,s_y)\rangle,\langle y,u_y\rangle\}$, $\gamma_{\sigma\varphi}=\{\langle x,u_x\rangle,\langle y,\lambda_\eta(u_y,s_y)\rangle\}$, $\mathcal{T}_{\sigma\varphi}=\{\langle x,t_x\rangle,\langle y,t_y+\tau_\eta(u_y,s_y)\rangle\}$, gdy wybierany jest element y i gdzie $\langle\{x,y\},\mathcal{N}_{\sigma\varphi}\rangle$ jest systemem wyboru wybierającym element dający minimum $\mathcal{T}_{\sigma\varphi}$.

Niech Σ_x będzie zbiorem wszystkich struktur $\sigma=\langle N_\sigma;\alpha_\sigma;\beta_\sigma,\gamma_\sigma,\mathcal{T}_\sigma\rangle$, gdzie $N_\sigma=\{x,y\}$, $\alpha_\sigma=\{\langle x,y\rangle\}$, $\beta_\sigma=\{\langle x,u_x\rangle,\langle y,u_y\rangle\}$, $\gamma_\sigma=\{\langle x,s_x\rangle\}$, $\mathcal{T}_\sigma=\{\langle x,t_x\rangle\}$, $u_x\in U$, $u_y\in U$, $s_x\in S$ i t_x jest rzeczywiste. Niech φ_x będzie funkcją przyporządkowującą każdej takiej strukturze strukturę $\sigma\varphi_x$, gdzie $N_{\sigma\varphi x}=N_\sigma$, $\alpha_{\sigma\varphi x}=\alpha_\sigma$, $\beta_{\sigma\varphi x}=\{\langle x,u_x\rangle,\langle y,\mu_\xi(u_x,s_x)\rangle\}$, $\gamma_{\sigma\varphi x}=\{\langle x,\lambda_\xi(u_x,s_x)\rangle\}$, $\mathcal{T}_{\sigma\varphi x}=\{\langle x,t_x+\tau_\xi(u_x,s_x)\rangle\}$.

Podobnie niech Σ_y będzie zbiorem wszystkich struktur $\sigma = \langle N_\sigma; \alpha_\sigma; \beta_\sigma, \gamma_\sigma, \mathcal{T}_\sigma \rangle$, gdzie $N_\sigma = \{x, y\}, \alpha_\sigma = \{\langle y, x\rangle\}$, $\beta_\sigma = \{\langle x, u_x\rangle, \langle y, u_y\rangle\}$, $\gamma_\sigma = \{\langle y, s_y\rangle\}$, $\mathcal{T}_\sigma = \{\langle y, t_y\rangle\}$, $u_x \in U$, $u_y \in U$, $s_y \in S$ i t_y jest rzeczywiste i niech φ_y będzie funkcją przyporządkowującą każdej takiej strukturze strukturę $\sigma\varphi_y$, gdzie $N_{\sigma\varphi_y} = N_\sigma$, $\alpha_{\sigma\varphi_y} = \alpha_\sigma$, $\beta_{\sigma\varphi_y} = \{\langle x, \mu_\eta(u_y, s_y)\rangle, \langle y, u_y\rangle\}$, $\gamma_{\sigma\varphi_y} = \{\langle y, \lambda_\eta(u_y, s_y)\rangle\}$, $\mathcal{T}_{\sigma\varphi_y} = \{\langle y, t_y + \tau_\eta(u_y, s_y)\rangle\}$.

Niech pr_x(pr_y) będzie funkcją przyporządkowującą każdej strukturze $\sigma \in \Sigma$ strukturę $\sigma pr_x \in \Sigma_x$, w której $N_{\sigma pr_x} = N_\sigma$, $\beta_{\sigma pr_x} = \beta_\sigma$ i $\alpha_{\sigma pr_x}$, $\gamma_{\sigma pr_x}$, $\mathcal{T}_{\sigma pr_x}$ są obcięciami α_σ, γ_σ, $\mathcal{T}_\sigma$ do $\{x\}$ (strukturę $\sigma_{pr_y} \in \Sigma_y$, w której $N_{\sigma pr_y} = N_\sigma$, $\beta_{\sigma pr_y} = \beta_\sigma$ i $\alpha_{\sigma pr_y}$, $\gamma_{\sigma pr_y}$, $\mathcal{T}_{\sigma pr_y}$ są obcięciami α_σ, γ_σ, $\mathcal{T}_\sigma$ do $\{y\}$).

Wówczas, przyjmując $\mathcal{N}(\sigma, \cdot) = \mathcal{N}_\sigma$, $\alpha(\sigma, \cdot) = \alpha_\sigma$, $\beta(\sigma, \cdot) = \beta_\sigma$, $\gamma(\sigma, \cdot) = \gamma_\sigma$, $\mathcal{T}(\sigma, \cdot) = \mathcal{T}_\sigma$ dla $\sigma \in \Sigma$, $\alpha_x(\sigma, \cdot) = \alpha_\sigma$, $\beta_x(\sigma, \cdot) = \beta_\sigma$, $\gamma_x(\sigma, \cdot) = \gamma_\sigma$, $\mathcal{T}_x(\sigma, \cdot) = \mathcal{T}_\sigma$ dla $\sigma \in \Sigma_x$ oraz $\alpha_y(\sigma, \cdot) = \alpha_\sigma$, $\beta_y(\sigma, \cdot) = \beta_\sigma$, $\gamma_y(\sigma, \cdot) = \gamma_\sigma$, $\mathcal{T}_y(\sigma, \cdot) = \mathcal{T}_\sigma$ dla $\sigma \in \Sigma_y$ mamy do czynienia z przypadkiem opisanym w[3] i $\langle \Sigma, \varphi\rangle$ jest maszyną będącą wiązką maszyn $\langle \Sigma_x, \varphi_x\rangle$, $\langle \Sigma_y, \varphi_y\rangle$ z funkcją wyboru $\sigma \to \langle Dom_{\mathcal{T}_\sigma}, \mathcal{N}_\sigma\rangle$ i z projekcjami pr_x, pr_y. Każdy przebieg maszyny $\langle \Sigma, \varphi\rangle$ jest równocześnie przebiegiem procesu współdziałania obu automatów.

Literatura

[1] DAHL O.J., NYGAARD K.: SIMULA - An ALGOL Based Simulation Language, Comm. of the ACM, Vol. 9, No 9, 1966.

[2] DAHL O.J., MYHRHAUG B., NYGAARD K.: SIMULA 67 - Common Base Language, Oslo 1968.

[3] PAWLAK Z.: Maszyny programowane, Algorytmy, Vol. V, No 10, 1969.

ДИСКРЕТНЫЕ ПРОЦЕССЫ ВЗАИМОДЕЙСТВИЯ

Резюме

В работе определена зависимость между машинами и их семействами. Эта зависимость используется в описании некоторых процессов взаимодействия.

DISCRETE INTERACTION PROCESSES

Summary

The paper defines a certain relation between machines and families of machines using it to describe some interaction processes.

Z.P.

ALGORYTMY VII, N° 13 1970, 17-34

PODGRAMATYKI ELEMENTARNE SKŁADNI ALGOLU 60

Jan MAŁUSZYŃSKI

Pracę złożono 15.6.1970

Praca zawiera definicję podgramatyki elementarnej gramatyki bezkontekstowej i opis składni ALGOLu 60 w postaci częściowo uporządkowanego zbioru podgramatyk elementarnych uzyskanych za pomocą maszyny ZAM 41. Opis ten ukazuje niektóre istotne cechy składni języka, które mogą być wyzyskane w programach analizy syntaktycznej oraz w nauczaniu języka. Metodę tę można stosować również do innych języków bezkontekstowych.

Gramatyki bezkontekstowe stosowane do opisu składni języków programowania zawierają zwykle dużą liczbę produkcji. Utrudnia to zrozumienie składni języka oraz analizę syntaktyczną. W pracy [3] podjęto próbę wyodrębnienia z gramatyki bezkontekstowej pewnych prostszych gramatyk bezkontekstowych, zwanych podgramatykami tej gramatyki. Ważną klasę takich podgramatyk stanowią podgramatyki elementarne. W tej pracy powtórzymy w sposób nieformalny definicję podgramatyki elementarnej oraz podamy krótki opis podgramatyk elementarnych składni języka ALGOL 60 uzyskanych za pomocą maszyny ZAM 41. Jako podstawę opisu gramatyki bezkontekstowej przyjmiemy w tej pracy notację BNF.

1. PODSTAWOWE POJĘCIA

Podając zbiór reguł BNF określających język musimy wskazać, które symbole mogą być z góry zadane jako początkowe elementy wywodów słów języka. Zwykle wyróżnia się w ten spo-

sób tylko jeden symbol i nazywa się go głową języka. W tej pracy nie będziemy ograniczali liczby wyróżnionych symboli, a ich zbiór nazwiemy *bazą*. Czasem wygodnie jest uważać za bazę wszystkie te symbole, które nie wchodzą w skład prawej strony żadnej reguły. W przypadku składni ALGOLu 60 są to symbole <program>, <number> i <basic symbol>.

Symbolami *nieterminalnymi* będziemy nazywać wszystkie te symbole, które występują jako lewe strony reguł BNF. Wszystkie pozostałe symbole uważać będziemy za symbole *terminalne*, np. <code> w ALGOLu 60.

Tworząc wywód, zastępujemy symbole nieterminalne występujące w elementach wywodu słowami określonymi przez reguły gramatyki. W tych słowach mogą również występować symbole nieterminalne, które będą eliminowane przy tworzeniu dalszych elementów wywodu. Każdy symbol nieterminalny gramatyki określa więc poprzez reguły gramatyki zbiór swoich *bezpośrednich następników* - tych symboli, które należą do prawej strony odpowiedniej reguły oraz zbiór swoich *następników* zawierający wszystkie te symbole, które mogą pojawić się w słowach wyprowadzonych z danego symbolu, a więc również i dany symbol. Jak widać, każdy symbol nieterminalny jest swoim następnikiem, lecz nie musi być swoim bezpośrednim następnikiem.

Ze zbioru wszystkich symboli nieterminalnych wyodrębnimy *klasy* - te podzbiory, w których relacja następstwa jest spójna i symetryczna. Z definicji tej wynika następująca podstawowa własność klasy symboli nieterminalnych: każdy symbol należący do klasy można przekształcić za pomocą reguł gramatyki w słowo zawierające dowolny inny symbol z tej klasy.

Jeżeli weźmiemy pod uwagę tylko wywody słów języka, to symbol nieterminalny może wystąpić w jakimś elemencie wywodu tylko wówczas, gdy należy do bazy gramatyki (pierwszy element wywodu) lub jest bezpośrednim następnikiem symbolu nieterminalnego występującego w jednym z poprzednich elementów wywo-

du. Dlatego w każdej klasie wyróżnimy te symbole, które należą do bazy gramatyki lub są następnikami elementów nie należących do klasy.

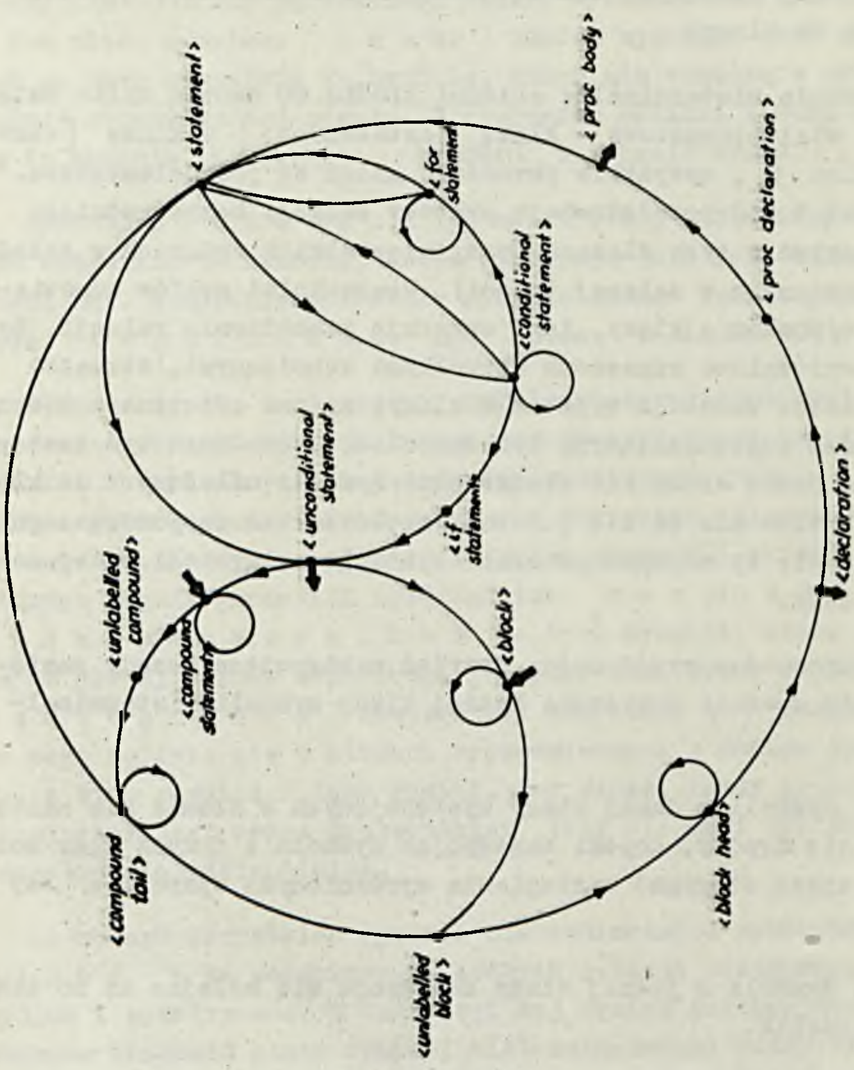
Symbole nieterminalne składni ALGOLu 60 tworzą tylko dwie klasy wieloelementowe - klasę [<statement>] i klasę [<expression>] , wszystkie pozostałe klasy są jednoelementowe. Rysunki 1 i 2 przedstawiają wykresy relacji bezpośredniego następstwa w tych klasach (przy niewielkich zmianach w składni omówionych w dalszej treści). Wierzchołki grafów odpowiadają symbolom z klasy, łuki wskazują zachodzenie relacji. Symbole wyróżnione oznaczono strzałkami wchodzącymi, strzałki wychodzące wskazują wyjścia z klasy; są one związane z wierzchołkami odpowiadającymi tym symbolom, które mogą być zastępowane przez słowa nie zawierające symboli należących do klasy. Takich słów nie da się już tak przekształcić za pomocą reguł gramatyki, by uzyskać ponownie wystąpienie symboli z tej samej klasy.

Konstruując wywód można przyjąć następujące zasady zastępowania symboli dotyczące każdej klasy symboli nieterminalnych:

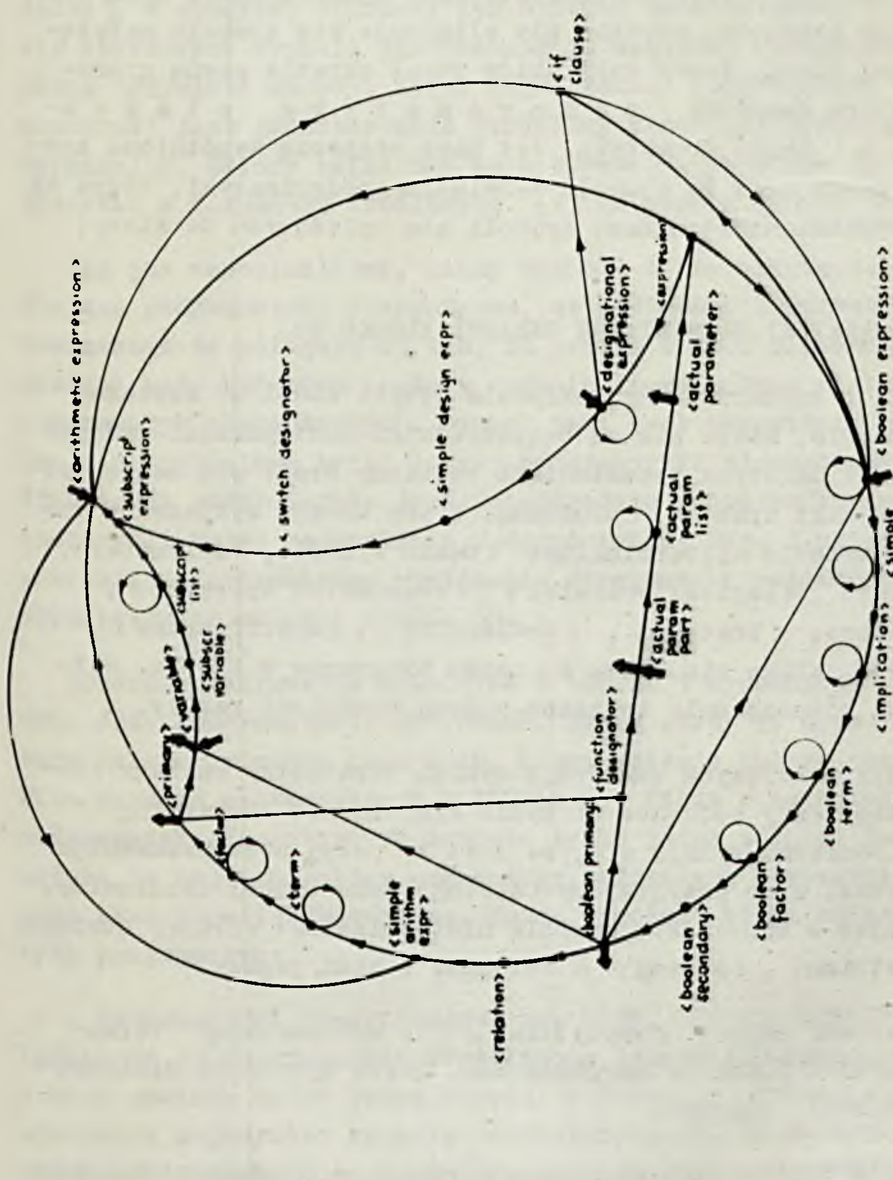
1. Symboli z danej klasy występujących w słowie nie zastępuje się dopóty, dopóki zastępując symbole z innych klas można jeszcze otrzymać wystąpienia wyróżnionych symboli z tej klasy.

2. Symbole z jednej klasy zastępuje się kolejno aż do ich wyczerpania.

Stosując te zasady mamy pewność, że symbole z wyczerpanych klas nie wystąpią w żadnym następnym elemencie wywodu. Ta metoda tworzenia wywodów nie ogranicza języka określanego przez gramatykę. Stosując ją w przypadku składni ALGOLu 60, uzyskiwalibyśmy np. w pewnym elemencie każdego wywodu wszystkie wystąpienia symbolu <identifier> zanim jakiekolwiek z nich zostałoby rozwinięte do konkretnego ciągu literowo-cyfrowego.



Rys. 1. Wykres relacji bezpośredniego następstwa dla klasy <statement>



Rys. 2. Wykres relacji bezpośredniego następstwa dla klasy <statement>

Jak widać, istnieje naturalna hierarchia klas ograniczająca kolejność zastępowania symboli nieterminalnych. Ta hierarchia jest częściowym porządkiem. Tworząc wywód według powyższych zasad, ze zbioru reguł związanych z każdą klasą korzysta się tylko lokalnie, wówczas gdy eliminuje się symbole należące do tej klasy. Każdy taki zbiór reguł określa pewną gramatykę, którą nazywamy p o d g r a m a t y k ą e l e m e n t a r n ą danej gramatyki. Jej bazę stanowią wyróżnione symbole z klasy oraz te symbole terminalne podgramatyki, które są bezpośrednimi następnikami symboli nie należących do klasy.

2. PODGRAMATYKI ELEMENTARNE SKŁADNI ALGOLU 60

Jak już wspominaliśmy, składnia języka ALGOL 60 zawiera trzy symbole, które nie są bezpośrednimi następnikami żadnych symboli. W dalszych rozważaniach będziemy brali pod uwagę tylko następniki symbolu <program>. W ten sposób wykluczamy ze składni symbole nieterminalne: <basic symbol>, <delimiter>, <operator> , <logical operator> , <sequential operator> , <separator>, <bracket> , <declarator> , <specificator> , <number> , które nie służą do opisu programów w ALGOLu. Eliminujemy równocześnie związane z tymi symbolami reguły.

Reguły metajęzyka zawierają symbol terminalny opisany nieformalnie: <any sequence of basic symbols not containing ´or`>. Ponieważ wydaje się, że jest to pewna niekonsekwencja metodyczna, więc przyjmujemy <string> jako symbol terminalny, eliminując w ten sposób symbole nieterminalne: <proper string>, <open string> , <string> i związane z nimi reguły.

Odrzucamy regułę <empty>::= ; wprowadzając terminalny symbol pusty ε wszędzie tam, gdzie występuje nieterminalny symbol <empty>.

Tabela 1 jest zestawieniem wszystkich podgramatyk elementarnych ograniczonej w ten sposób składni ALGOLu 60. Pierwsza kolumna tabeli zawiera kolejne liczby, druga – umowne nazwy

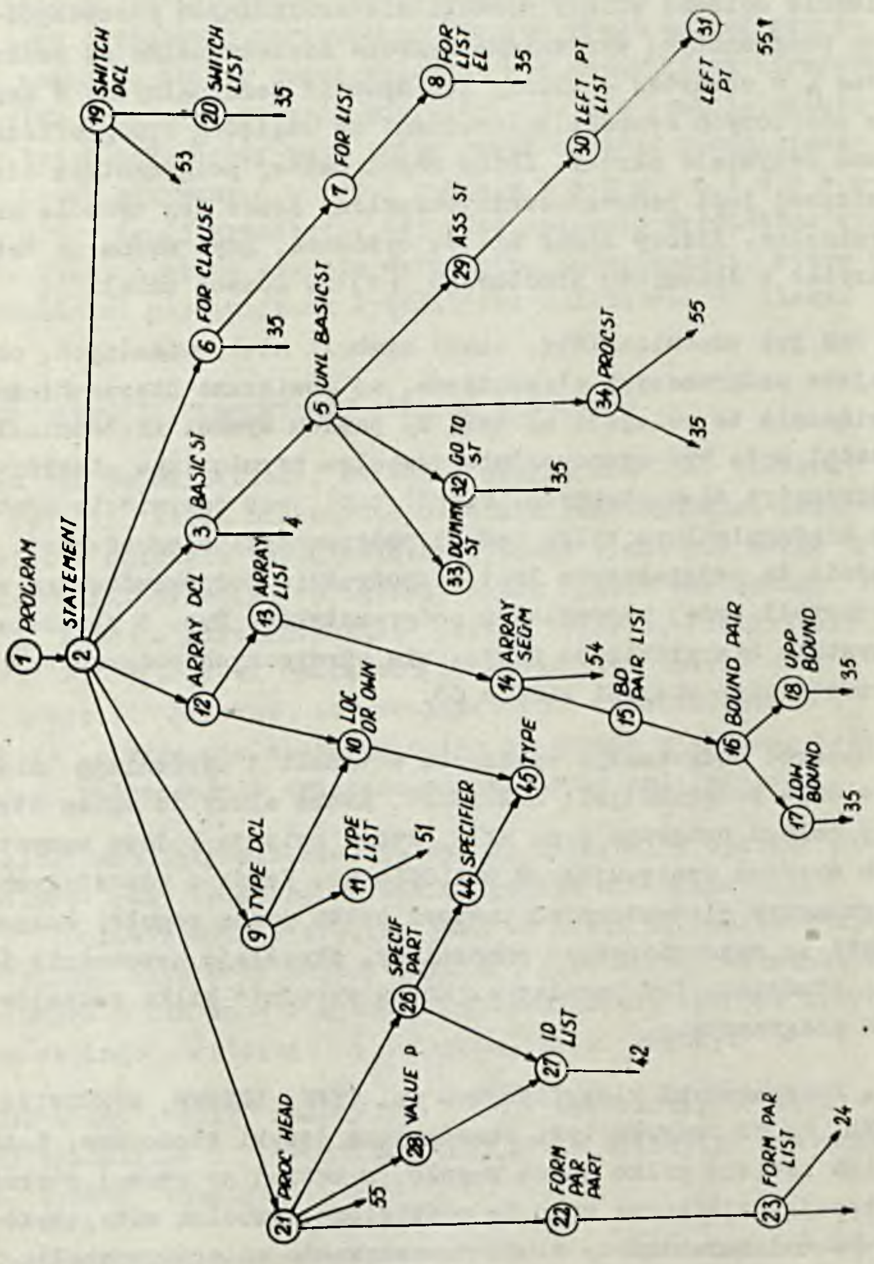
podgramatyk. Oznaczenia te są stosowane na rys. 3. W trzeciej kolumnie opisano zbiory symboli nieterminalnych poszczególnych podgramatyk (wyróżnione symbole nieterminalne są podkreślone), w czwartej - zbiory ich symboli terminalnych. W zapisie niektórych symboli zastosowano ze względów typograficznych pewne oczywiste skróty. Zbiór reguł każdej podgramatyki elementarnej jest jednoznacznie określony przez jej symbole nieterminalne. Zbiory reguł nie są cytowane, gdyż można je łatwo odczytać z dokumentu źródłowego [1] za pomocą tabeli 1.

Jak już wspominaliśmy, klasy symboli nieterminalnych, określające podgramatyki elementarne, są powiązane hierarchicznie. Powiązania te polegają na tym, że pewien symbol nieterminalny składni może być równocześnie symbolem terminalnym niektórych podgramatyk elementarnych. Symbol taki jest oczywiście symbolem nieterminalnym tylko jednej podgramatyki elementarnej. Właśnie ta podgramatyka jest bezpośrednio podporządkowana w hierarchii wyżej wspomnianym podgramatykom. Rys. 3 przedstawia wszystkie hierarchiczne powiązania otrzymanych podgramatyk elementarnych składni ALGOLu 60.

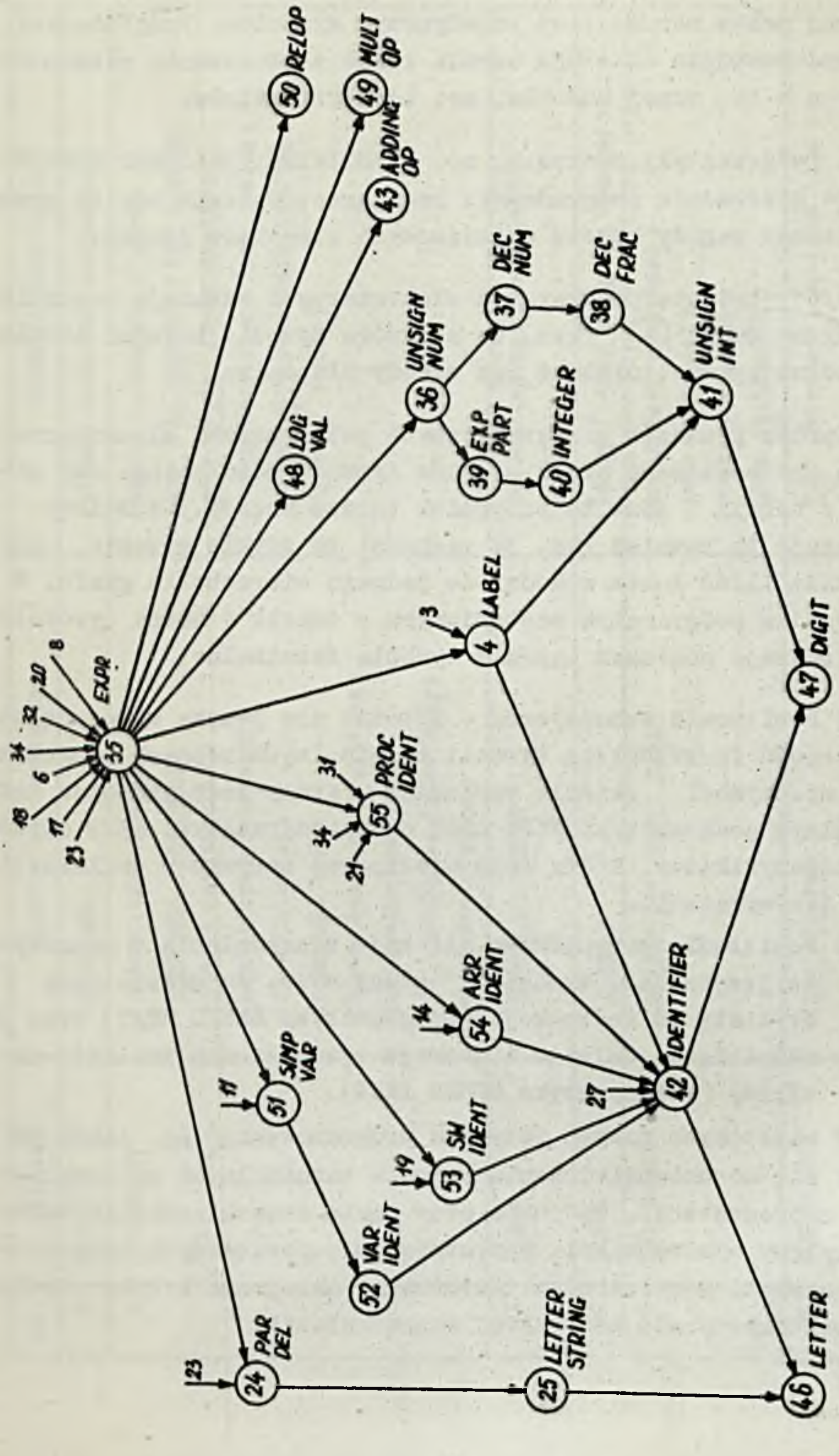
Spośród podgramatyk opisanych w tabeli 1 wyróżniają się dwie duże podgramatyki: STATEMENT, która służy do opisu struktury całego programu oraz EXPR, która opisuje budowę wszystkich wyrażeń występujących w ALGOLu 60. Każda z pozostałych podgramatyk elementarnych zawiera tylko jedną regułę. Podgramatyki te mają charakter pomocniczy, określają przeważnie języki skończone lub regularne. Można wyróżnić kilka rodzajów tych podgramatyk:

. Podgramatyki klasyfikujące np. TYPE, LETTER, SPECIFIER, LABEL. Są to podgramatyki określające języki skończone. Każda z nich zawiera tylko jedną regułę, w której po prawej stronie występują pojedyncze symbole oddzielone symbolem metajęzykowym |. Podgramatyki te służą do nazywania zbiorów symboli.

. Podgramatyki przemianowujące np. VAR IDENT, SWITCH IDENT. Podgramatyka przemianowująca zawiera tylko jedną regułę, w



Rys. 3.1. Hierarchiczne powiązania podgramatyk ALGOLu 60



Rys. 3.2. Hierarchiczne powiązania podgramatyk ALGOLu 60

której prawa strona jest pojedynczym symbolem. Podgramatyki przemianowujące wskazują zwykle różne zastosowania elementów języka o tej samej budowie, np. identyfikatorów.

. Podgramatyki tworzące, np. IDENTIFIER, UNSIGNED NUMBER. Są to przeważnie podgramatyki regularne. Opisują one za pomocą jednej reguły budowę podstawowych elementów języka.

Wyodrębnienie podgramatyk elementarnych wskazuje naocznie znany od dawna [4] fakt, że niektóre symbole składni ALGOLu 60 można by wyeliminować bez szkody dla opisu.

Oprócz powiązań hierarchicznych podgramatyki elementarne mogą być powiązane przez wspólne symbole terminalne. Jak widać z tabeli 1 jest to przypadek bardzo częsty. Częściowo obrazuje to również rys. 3; zachodzi to zwykle wówczas, gdy większa ilość łuków wchodzi do jednego wierzchołka grafu. W przypadku podgramatyk zestawionych w tabeli 1 można wyróżnić dwa rodzaje powiązań poprzez symbole terminalne:

. Powiązania semantyczne - istotne dla języka wskazują na tożsamość funkcjonalną symboli terminalnych różnych podgramatyk np. symbol <simple variable> , który jest symbolem terminalnym podgramatyki TYPE LIST oraz podgramatyki EXPR określa identyfikator, który może występować zarówno w deklaracji jak i w wyrażeniu.

. Powiązania przypadkowe nie mają uzasadnienia w semantyce języka, np. symbol ":" służy m.in. do oddzielania etykiety od instrukcji (podgramatyka BASIC STAT) oraz do oddzielania dolnego i górnego ograniczenia indeksów macierzy (podgramatyka BOUND PAIR).

W niektórych nowych językach programowania(np. ALGOL 68) dąży się do uniezależnienia symboli terminalnych od konkretnej reprezentacji. Być może przy takim uniezależnieniu celowe byłoby wyodrębnienie wszystkich nie powiązanych semantycznie symboli terminalnych. Ewentualne sklejenia byłyby cechą charakterystyczną konkretnej reprezentacji.

Tabela 1

Lp	Nazwa	Symbole nieterminalne	Symbole terminalne
1	PROGRAM	<program>	<block> <compound statement>
2	STATEMENT	<unconditional statement> <compound statement> <block> <statement> <conditional statement> <for statement> <compound tail> <block head> <declaration> <unlabelled compound> <if statement> <proc body> <proc declaration>	<basic statement> <label> : else <if clause> <for clause> end ; begin <type declar> <array declar> <switch declar> procedure <procedure heading> <type> , <code>
3	BASIC	<basic statement>	<unlabelled basic statement> <label> :
4	LABEL	<label>	<identifier> <unsigned integer>
5	UNL BASIC ST	<unlabelled basic stat>	<assignment statement> <goto statement> <dummy statement> <proc statement>
6	FOR CLAUSE	<for clause>	for <variable> := <for list> do

Tabela 1 /c.d./

Lp	Nazwa	Symbole nieterminalne	Symbole terminalne
7	FOR LIST	<for list>	<for list element> ,
8	FOR LIST EL	<for list element>	<arithmetic expr> step until while <boolean expr>
9	TYPE DCL	<type declar>	<local or own type><type list>
10	LOC OR OWN	<local or own type>	<type> own
11	TYPE LIST	<type list>	<simple variable> ,
12	ARRAY DCL	<array declar>	array <array list> <local or own type>
13	ARRAY LIST	<array list>	<array segment> ,
14	ARRAY SEGM	<array segment>	<bound pair list> [ ] <array identifier>
15	BD PAIR LIST	<bound pair list>	<bound pair> ,
16	BOUND PAIR	<bound pair>	<lower bound><upper bound> :
17	LOW BOUND	<lower bound>	<arithmetic expr>
18	UPP BOUND	<upper bound>	<arithmetic expr>
19	SWITCH DCL	<switch declar>	<switch list> switch <switch identifier> :=

Tabela 1 /c.d./

Lp	Nazwa	Symbole nieterminalne	Symbole terminalne
20	SWITCH LIST	<switch list>	<designational expr> ,
21	PROC HEAD	<procedure heading>	<formal parameter part> <specification part> <value part> ; <procedure identifier>
22	FORM PAR PART	<formal parameter part>	<formal parameter list> ε ( )
23	FORM PAR LIST	<formal parameter list>	<formal parameter> <parameter delimiter>
24	PAR DEL	<parameter delimiter>	<letter string> , : ( )
25	LETTER STRING	<letter string>	<letter>
26	SPECIF PART	<specification part>	<specifier> <identifier list> ; ε
27	ID LIST	<identifier list>	<identifier> ,
28	VALUE P	<value part>	<identifier list> value ; ε
29	ASS ST	<assignment statement>	<left part list> <boolean expr> <arithmetic expr>
30	LEFT PT LIST	<left part list>	<left part> ,

Tabela 1 /c.d./

Lp	Nazwa	Symbole nieterminalne	Symbole terminalne
31	LEFT PT	<left part>	<variable> <proc identifier> :=
32	GOTO ST	<goto statement>	<designational expr> goto
33	DUMMY ST	<dummy statement>	ε
34	PROC ST	<procedure statement>	<actual parameter part> <procedure identifier>
35	EXPR	<expression> <arithmetic expr> <boolean expr> <designational expr> <subscript expr> <subscript list> <subscripted variable> <variable> <actual param> <actual parameter list> <actual parameter part> <function desig> <primary> <factor><term><if clause> <simple arithmetic expr> <relation><bool primary> <boolean secondary>	else , <array identifier> [ ] <simple variable> if then <string> <switch ident> <procedure identifier> <parameter delimiter> ( ) <unsigned number> ↑ <multiplying operator> <adding operator><label> <relational operator> <logical value> ≡ ∧ ∨ ¬ → ε

Tabela 1 /c.d./

Lp	Nazwa	Symbole nieterminalne	Symbole terminalne
		<boolean factor> <bool term> <implication> <simple bool> <switch designator> <simple designational expr>	
36	UNSIGN NUM	<unsigned number>	<decimal number> <exponent part>
37	DEC NUM	<decimal number>	<unsigned integer> <decimal fraction>
38	DEC FRAC	<desimal fraction>	<unsigned integer> .
39	EXP PART	<exponent part>	<integer> $_{10}$
40	INTEGER	<integer>	<unsigned integer> + −
41	UNSIGN INT	<unsigned integer>	<digit>
42	IDENTIFIER	<identifier>	<letter> <digit>
43	ADDING OP	<adding operator>	+ −
44	SPECIFIER	<specifier>	string array switch procedure <type>
45	TYPE	<type>	real integer boolean
46	LETTER	<letter>	a b c d e f g h i j k l m n o p q r s t u v w x

Tabela 1 /c.d./

Lp	Nazwa	Symbole nieterminalne	Symbole terminalne
			y z A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
47	DIGIT	<digit>	1 2 3 4 5 6 7 8 9 0
48	LOG VAL	<logical value>	true false
49	MULT OP	<multiplying operator>	x / -
50	REL OP	<relational operator>	= ≤ < > ≥ ≠
51	SIMP VAR	<simple variable>	<variable identifier>
52	VAR IDENT	<variable identifier>	<identifier>
53	SW IDENT	<switoh identifier>	<identifier>
54	ARR IDENT	<array identifier>	<identifier>
55	PROC IDENT	<procedure identifier>	<identifier>

3. ZAKOŃCZENIE

Wyodrębnienie podgramatyk elementarnych składni ALGOLu 60 pozwoliło wskazać w sposób naoczny niektóre istotne cechy tej składni. Ujawniła się szczególna rola niektórych symboli nieterminalnych oraz hierarchiczne powiązania podgramatyk. Znajomość tych cech może być użyteczna przy analizie syntaktycznej programów [2]. Podgramatyki elementarne mogą również stanowić cenną pomoc metodyczną ułatwiającą systematyczny wykład języka. Powiązania podgramatyk poprzez symbole terminalne ułatwiają zrozumienie semantyki języka.

Program wydzielania podgramatyk elementarnych, za pomocą którego uzyskano powyższe wyniki może być zastosowany do dowolnego zbioru reguł zapisanych w postaci BNF.

Literatura

[1] NAUR P. ed.: Revised Report on the Algorithmic Language ALGOL 60. Comm. ACM 6 1963, pp. 1-17.

[2] MAŁUSZYŃSKI J.: Analiza syntaktyczna przy użyciu separatorów. Prace I Ogólnokrajowej Konferencji np. "Naukowe problemy maszyn matematycznych". PWN, w druku.

[3] MAŁUSZYŃSKI J.: Algebraiczne własności podgramatyk bezkontekstowych i analiza syntaktyczna. Prace COPAN, w przygotowaniu.

[4] CULIK K.: Formal Structure of ALGOL and Simplification of its Description. Symbolic Languages in Data Processing. Gordon and Breach. New York. 1962, pp. 75-82.

ЭЛЕМЕНТАРНЫЕ ПОДГРАММАТИКИ СИНТАКСИСА ЯЗЫКА АЛГОЛ-60

Работа содержит определение элементарной подграмматики бесконтекстной грамматики и описание синтаксиса языка АЛГОЛ-60 в виде частично упорядоченного множества элементарных подграмматик полученных на машине ZAM-41. Это описание указывает некоторые существенные синтаксические свойства языка, которые могут быть использованы в программах синтаксического анализа и при обучении языка. Этот метод применим и для других бесконтекстных грамматик.

THE ELEMENTARY SUBGRAMMARS OF ALGOL 60 SYNTAX

Summary

The paper contains the definition of elementary subgrammar of a CF-grammar and the description of the ALGOL 60 syntax, as the partially ordered set of the elementary subgrammars received on ZAM-41 computer. This description points out some important syntactic properties of the language, which can be useful in parsing technique, and in the programming language teaching. This method can be applied for other CF-grammars.

ALGORYTMY, VII, N° 13 1970, 35-43

COMPOSITION OF PUSHDOWN TRANSDUCERS

by Marek GŁOWACKI
Received May 5th,1970

In this paper, we consider mappings being superpositions of finite number pushdown transducer mappings. Such mapping can be considered as a mathematical model of multipass compiler. It is shown that such compiler cannot "realize" arbitrary recursive mappings.

LIST OF SYMBOLS

$X \cup Y$	The union of sets X, Y
$X \times Y$	The cartesian product of sets X, Y
$\lvert X \rvert$	The cardinal number of X
$\emptyset$	The empty set
A^*	The set of all words over an alphabet A
ϵ	The empty word
A^+	The set of all, nonempty words over A
$\lvert w \rvert$	The length of word w

1. INTRODUCTORY CONCEPTS

In certain data-processing situations, an output structure is sometimes added to a pushdown automaton. The automaton so obtained is called a pushdown transducer. Mappings defined by pushdown transducers have been studied in [1].

In this paper, we consider mappings being superpositions of finite number pushdown transducer mappings.

DEFINITION. A pushdown transducer (abbreviated pdt) is a 7-tuple $A = (Q, I, T, O, \mu, Z_0, q_0)$, where:

(i) Q is a finite, nonempty set (of states).
(ii) I is an alphabet (of inputs).
(iii) T is a finite, nonempty set (of pushdown symbols).

(iv) O is an alphabet (of outputs).

(v) μ is a mapping of $Q \times (I \cup \{\epsilon\}) \times T$ into the finite subsets of $Q \times T^* \times O^*$.

(vi) Z_0 is an element of T (the start pushdown symbol).

(vii) q_0 is in Q (the start state).

We now introduce symbolism enabling us to discuss the outputs from pdt.

Given a pdt $A = (Q, I, T, O, \mu, Z_0, q_0)$, let $\overset{*}{\underset{A}{\vdash}}$ (or $\overset{*}{\vdash}$ if A is understood) be the transitive closure of the binary relation $\vdash$ on $Q \times I^* \times T^* \times O^*$ defined as follows. For Z in T and x in $I \cup \{\epsilon\}$ let $(p, xw, \alpha Z, y) \vdash (q, w, \alpha\gamma, yv)$ if (q, γ, v) is in the set $\mu(p, x, Z)$.

For each x in I^* let A(x) be the set of all words y in O^* such that $(q_0, x, Z_0, \epsilon) \overset{*}{\vdash} (q, \epsilon, \alpha, y)$ for some (q, α) in $Q \times T^*$.

The mapping of x into A(x) for each x in I^* is called a pdt mapping.

DEFINITION. A composition of pdt mappings A and B is the mapping defined as follows. For x in the input alphabet of A

$$B \cdot A(x) = \bigcup_{y \text{ in } A(x)} B(y).$$

A composition of three and more pdt's is defined analogously.

It has been shown by Ginsburg and Rose [1] that the composition of pdt mappings need not be a pdt mapping.

2. PROPER PUSHDOWN TRANSDUCERS

DEFINITION. A pdt A is said to be proper if for each input x the set A(x) is finite.

A composition of finite number of proper pdt mappings can be considered as a mathematical model of compiler. In this paper it is shown that such compiler cannot "realize" arbitrary recursive mapping but only such which is computable by linear bounded transducer.

A composition of two proper pdt mappings is not necessarily a pdt mapping. For example, let A and B be proper pdt's defined as follows:

$A = (Q, \{a, b, c\}, \{Z, a\}, \{a, c, r\}, \delta, Z, s)$, where
$Q = \{q, p, r, s\}$ and for x in $\{a, b, c\}$, S in $\{Z, a\}$,
$\delta(s, a, Z) = \{(q, Za, \epsilon\}$, $\delta(q, a, a) = \{(q, aa, \epsilon)\}$,
$\delta(q, b, a) = \delta(p, b, a) = \{(p, \epsilon, a)\}$, $\delta(q, c, S) =$
$= \{(r, Z, r)\}$, $\delta(r, x, S) = \{(r, S, \epsilon)\}$,
$B = (K, \{a, c, r\}, \{Z, a\}, \{0,1\}, \mu, Z, q)$, where
$K = \{q, r, p\}$ and for x in $\{a, c, r\}$, S in $\{Z, a\}$,
$\mu(q, a, S) = \{(q, Sa, \epsilon)\}$, $\mu(q, r, S) = \mu(q, c, Z) =$
$= \{(r, Z, 0)\}$,
$\mu(q, c, a) = \{(q, \epsilon, \epsilon), (p, \epsilon, \epsilon)\}$, $\mu(p, \epsilon, Z) =$
$= \{(p, \epsilon, 1)\}$,
$\mu(p, \epsilon, a) = \{(r, \epsilon, 0)\}$, $\mu(r, x, Z) = \{(r, Z, \epsilon)\}$.

Clearly, for x in $\{a, b, c\}^*$ $A(x) = a^i c^j$ if $x = a^i b^i c^j$ $(i, j = 1, 2, \ldots)$ and $A(x) = yr$, with y in $\{a, c\}^*$, or is undefined in the remaining cases.

For u in $\{a, c, r\}^*$ $B(u) = 1$, if $u = a^i c^i$ $(i=1,2,\ldots)$ and $B(u) = 0$ in the remaining cases. Then the composition $B \circ A$ is the characteristic function of the set $\{a^i b^i c^i :$ $i = 1, 2, \ldots\}$, which is not context free. It easy implies that $B \circ A$ is not a pdt mapping.

NOTATION. Let $A = (Q, I, T, O, \delta, Z_0, q_0)$ be a pdt, and let n be an integer. Write $(q, x, \alpha Z, y) \vdash^{n} (p, u, \gamma, yv)$, if $(q, x, \alpha Z, y) \vdash (p_1, x_1, \gamma_1, yv_1) \vdash \ldots \vdash$ $(p_k, x_k, \gamma_k, yv_1 \ldots v_k) = (p, u, \gamma, yv)$, and $|\gamma_i| \leq n$, for $i = 1, \ldots, k$.

LEMMA 1. For each proper pdt $A = (Q, I, T, O, \delta, Z_0, q_0)$ there exists a pdt $B = (Q, I, T, O, \mu, Z_0, q_0)$ such that:

(1) $A(x) = B(x)$, for each x in I^+.

(2) For each (q, Z) in $Q \times T$ if (p, α, y) is in $\mu(q, \epsilon, Z)$, then $\alpha = \epsilon$.

Proof: Without loss generality we may assume that $|\alpha| \leqslant 2$ and $p \neq q_0$ whenever $\delta(q, s, Z)$ contains (p, α, y).

Let $n = |Q| \cdot |T|$. We define μ as follows. For (q, Z) in $Q \times T$ and x in I let:

(i) $\mu(q, \epsilon, Z) = \{(p, \epsilon, y) : (q, \epsilon, Z, \epsilon) \vdash_A^n (p, \epsilon, \epsilon, y)\}$

(ii) for $(q, Z) \neq (q_0, Z_0)$,
$\mu(q, x, Z) = \{(p, \gamma, yv) : (r, \epsilon, \alpha S, \epsilon) \vdash_A^{n+1} (p, \epsilon, \gamma, v)$,
for $(r, \alpha S, y)$ in $\delta(q, x, Z)\}$,

(iii) $\mu(q_0, x, Z_0) = \{(p, \alpha\gamma, yv) : (q_0, \epsilon, Z_0, \epsilon) \vdash_A^n (r, \epsilon, \alpha Z, y)$,
and (p, γ, v) is in $\mu(r, x, Z)\}$.

Consider a computation $(q, \epsilon, SZ, \epsilon) \vdash_A^* (p, \epsilon, Z_0 Z_1 \ldots Z_k, y)$.

There are states $p_1, \ldots, p_k$ in Q and $y_1, \ldots, y_k$ in O^* such that $(q, \epsilon, SZ, \epsilon) \vdash^* (p_1, \epsilon, Z_0 Z_1, y_1) \vdash^*$
$(p_2, \epsilon, Z_0 Z_1 Z_2, y_1 y_2) \vdash^* \ldots \vdash^* (p_k, \epsilon, Z_0 \ldots Z_k, y_1 \ldots y_k) =$
$= (p, \epsilon, Z_0 \ldots Z_k, y)$.

Suppose that $k > n$. Then there exist i, j $(1 \leqslant i < j \leqslant k)$ such that $(p_i, Z_i) = (p_j, Z_j)$, and we have:

$(p_i, \epsilon, Z_0 \ldots Z_j, \epsilon) \vdash^* (p_j, \epsilon, Z_0 \ldots Z_i (Z_{i+1} \ldots Z_j)^m,$
$(y_{i+1} \ldots y_j)^m)$ for each $m = 0, 1, \ldots$ It may be the following cases:

a. $y_{i+1} \ldots y_j \neq \epsilon$. Then

$(q, \epsilon, SZ, \epsilon) \vdash^* (p_i, \epsilon, Z_0 \ldots Z_i (Z_{i+1} \ldots Z_j)^m,$

$(y_{i+1} \ldots y_j)^m)$,

for each $m = 0, 1, \ldots$; and, because A is proper, for each x such that $A(x)$ is defined, A having x as input cannot attain any configuration with q and SZ as a state and top of the stack respectively.

b. $y_{i+1} \dots y_j = \epsilon$. Then there exists a computation

$(q, \epsilon, SZ, \epsilon) \overset{*}{\underset{A}{\vdash}} (p_1, \epsilon, Z_0Z_1, y_1) \overset{*}{\underset{A}{\vdash}} (p_k, \epsilon, Z_0Z_1 \dots Z_iZ_{j+1}, y_1 \dots y_iy_{j+1} \dots y_k)$, where $y_1 \dots y_iy_{j+1} \dots y_k = y$. Now, if $k - 1 > n$, we can reduce some Z_i by repeating the above procedure, otherwise if (q, SZ, v) is in (r, x, R), for some (r, x, R) in $Q \times I \times T$, then $(p_k, Z_0 \dots Z_iZ_{j+1} \dots Z_k, vy)$ is in $\mu(r, x, R)$.

Let $x = x_1 \dots x_m$ be in I^+ and let $(q_0, x, Z_0, \epsilon) \overset{*}{\underset{A}{\vdash}} (p, \epsilon, \gamma, y)$ for some (p, γ) in $Q \times T^*$. This computation is of the form:

$$\begin{aligned}
&(q_0, x_1 \dots x_m, Z_0, \epsilon) \overset{*}{\underset{A}{\vdash}} (q_1, x_1 \dots x_m, \alpha_1 S_1, y_1) \underset{A}{\vdash} \\
&\quad (r_1, x_2 \dots x_m, \alpha_1' S_1', y_1 y_1') \overset{*}{\underset{A}{\vdash}} \\
&\quad \overset{*}{\underset{A}{\vdash}} (p_1, x_2 \dots x_m, \alpha_1'' S_1'', y_1 y_1' y_1'') \overset{*}{\underset{A}{\vdash}} \dots \\
&\quad \overset{*}{\underset{A}{\vdash}} (q_i, x_i \dots x_m, \alpha_i S_i, y_1 \dots y_i) \underset{A}{\vdash} \\
&\quad \underset{A}{\vdash} (r_i, x_{i+1} \dots x_m, \alpha_i' S_i', y_1 \dots y_i y_i') \overset{*}{\underset{A}{\vdash}} \\
&\quad \overset{*}{\underset{A}{\vdash}} (p_i, x_{i+1} \dots x_m, \alpha_i'' S_i'', y_1 \dots y_i y_i' y_i'') \overset{*}{\underset{A}{\vdash}} \\
&\quad \overset{*}{\underset{A}{\vdash}} (p, \epsilon, \gamma, y), \quad \text{for } i = 1, \dots, m.
\end{aligned}$$

Observe that by the above remarks:

$(p_1, \alpha_1'' S_1'', y_1 y_1' y_1'')$ is in $\mu(q_0, x_1, Z_0)$, and for $2 \leq i \leq m$,

$(p_i, \alpha_i'' S_i'', y_i y_i' y_i'')$ is in $\mu(q_i, x_i, S_i)$.

Since, for each x in I^*, if $(q_0, x, Z_0, \epsilon) \overset{*}{\underset{A}{\vdash}}$ $\overset{*}{\underset{A}{\vdash}} (p, \epsilon, \gamma, y)$, then $(q_0, x, Z_0, \epsilon) \overset{*}{\underset{B}{\vdash}} (p, \epsilon, \gamma, y)$ and the theorem holds.

LEMMA 2. For each proper pdt $A = (Q, I, T, O, \delta, Z_0, q_0)$ there exists a pdt $B = (Q, I, T, O, \mu, Z_0, q_0)$, and an integer b such that $A(x) = B(x)$, for all x in I^+ and:

(3) B computes B(x) by a sequence of moves in which the stack never grows longer than $b \cdot |x|$.

Proof: Let B be the pdt in Lemma 1 and let
$b = \max\{|\alpha| - 1 : (p, \alpha, y)$ is in (q, s, Z), for (q, s, Z) in $Q \times I \times T\}$.

If y is in B(x), where $x = x_1 \dots x_m$ (x_i in I, for $1 \leqslant i \leqslant m$) then there exists a computation:

$$(q_0, x_1 \dots x_m, Z_0, \epsilon) \vdash_B (q_1, x_2 \dots x_m, \gamma_1, y_1) \vdash^*$$
$$(p_1, x_2 \dots x_m, \gamma_1', y_1') \vdash^* (q_i, x_{i+1} \dots x_m, \gamma_i, y_i) \vdash^*$$
$$(p_i, x_{i+1} \dots x_m, \gamma_i', y_i') \vdash (q_{i+1}, x_{i+2} \dots x_m, \gamma_{i+1}, y_{i+1})$$
$$\vdash^* \dots \vdash^* (p_m, \epsilon, \gamma_m', y_m'), \quad \text{where } y_m' = y.$$

Since B satisfies (2), then $|\gamma_i'| \leqslant |\gamma_i|$, for $i = 0, \dots, m$ and by definition of b, for each i $|\gamma_i| \leqslant b \cdot m = b \cdot |x|$.

3. PUSHDOWN TRANSDUCERS AND LINEAR BOUNDED TRANSDUCERS

DEFINITION. A linear bounded transducer (abbreviated lbt) is a 5-tuple $A = (Q, I, O, \delta, q_0)$, where

(i) Q, I, O and q_0 have the same significance as in a pdt.

(ii) δ is a mapping of $Q \times I$ into the finite subsets of $Q \times I \times O^* \times \{-1, 0, 1\}$.

Given an lbt $A = (Q, I, O, \delta, q_0)$, let $\vdash_A^*$ be the transitive closure of the binary relation $\vdash$ on $I^*QI^* \times O^*$ defined as follows:

For u, v in I^*, c in I and w in O^*:

(i) $(ucpav, w) \vdash (uqcbv, wy)$ if $(q, b, y, -1)$ is in $\delta(p, a)$,

(ii) $(upav, w) \vdash (uqbv, wy)$ if $(q, b, y, 0)$ is in $\delta(p, a)$,

(iii) $(upav, w) \vdash (ubqv, wy)$ if $(q, b, y, 1)$ is in $\delta(p,a)$.

For each lbt $A = (Q, I, O, \delta, q_o)$, and each word w in I^* let $A(w) = \{y \text{ in } O^* : (q_o\ w, \epsilon) \overset{*}{\underset{A}{\vdash}} (\beta, y)$, for some β in $I^*Q\}$.

In particular, $A(\epsilon) = \{\epsilon\}$.

Given an lbt A, the function $A(x)$ of I^* into the subsets of O^* is called an lbt mapping.

THEOREM. For arbitrary proper pdt's A_1, A_2 there exists an lbt B such that $B(x) = A_2 \circ A_1(x)$, for each x in I^+.

<u>Proof:</u> Let $A_i = (Q_i, I_i, T_i, O_i, \mu_i, Z_{oi}, q_{oi})$ be a proper pdt for i = 1, 2. By Lemma 2 we can assume that there exist integers b_1, b_2 such that for A_i, b_i the condition (3) holds. Let $k = \max\{|y| : (p, \alpha, y) \text{ is in } \mu_1(q, x, Z)\}$.

We shall construct a fourtape Turing machine M computing $A_2 \circ A_1$. The machine consists of a finite control and four tapes. On tape 1, M will hold an input word. Pushdown stores of A_1 and A_2 will be simulated by M on tapes 2 and 3 respectively. Tape 4 of the fixed length k will hold words being produced by A_1 one by one.

Let x be an input word. Initially, the finite control of M is in the designated "start state", with the input on tape 1 and Z_{o1}, Z_{o2} on tape 1 and 2 respectively. All remaining cells of all tapes hold the special symbol b symbolizing blank.

M computes $A_2 \circ A_1$ as follows:

I. M simulates A_1 using tape 2 as the pushdown store of A_1 until some nonempty word is obtained as a part of output. Then the word so obtained is stored on tape 4 and:

II. M simulates A_2 using tape 3 as the pushdown store and the word on tape 4 as input until the word on tape 4 becomes empty (i.e. there are blank symbol only). Then M repeats I

provided there are nonblank symbols on tape 1. Otherwise, the computation is ended.

By Lemma 2 there exists a computation in which tapes 2 and 3 never grows longer than $b_1 \cdot |x|$ and $b_2 \cdot k \cdot |x|$ respectively. It is trivial to show that for an m-tape Turing machine of tape complexity $L(n) = k \cdot n$ (k is an integer) there exists an lbt realizing the same mapping. If I is the alphabet of M, it suffices to extend this alphabet to the set:

$$I \cup \{ (w_1, \ldots, w_m) \text{ in } (I^*)^m : |w_i| < k, \text{ for } i = 1, \ldots, m \}$$

The lbt, having this alphabet, will be simulate i-th tape of M on i-th "coordinate" - if $(w_{11}, \ldots, w_{1m}) \ldots (w_{n1}, \ldots, w_{nm})$ will be a word on the tape, then $w_{1i} \ldots w_{ni}$ will be represent contents of i-th tape of M. Then, the machine constructed above may be an lbt.

COROLLARY. For each sequence $A_1, \ldots, A_n$ of proper pdt's there exists an lbt B such that $B(x) = A_n \circ \ldots \circ A_1(x)$, for each nonempty input x.

Proof: The proof is the same as in the Theorem but we construct $(2n + 1)$-tape Turing machine.

REMARK. In this paper we have studied only proper pdt mappings. In general, there are pdt mappings which are not lbt mappings. For example, see [1].

References

[1] GINSBURG S. and ROSE G.F. : Preservation of Languages by Transducers. Information and Control vol. 9, 153-176, 1966.

KOMPOZYCJA TRANSLATORÓW STOSOWYCH

Praca dotyczy pewnych maszyn abstrakcyjnych zwanych translatorami stosowymi. Złożenie odwzorowań określonych przez takie maszyny może być traktowane jako matematyczny model wieloprzebiegowego kompilatora. Pokazano, że kompilator taki nie może "realizować" dowolnych przekształceń rekurencyjnych, a tylko niektóre z nich.

СОСТАВЛЕНИЕ СТЕКОВЫХ ТРАНСЛЯТОРОВ

Резюме

Настоящий доклад представляет некоторый класс абстрактных машин называемых стэковыми трансляторами. Сложение преобразований определенных такими машинами можно рассматривать как математическую модель многопрогонного компилятора. В работе показывается, что такой компилятор не может "реализовать" произвольных рекурсивных преобразований а только некотории из них.

ALGORYTMY VII, N° 13 1970, 45-70

RÓWNANIA I NIERÓWNOŚCI ALGEBRAICZNE Z NIEWIADOMYMI BINARNYMI

Marian LIGMANOWSKI

Pracę złożono 18.10.1968

Podano dwie metody rozwiązywania równań i nierówności algebraicznych, zwanych też pseudobulowskimi, dostosowane do obliczeń za pomocą maszyn cyfrowych. Pierwsza jest udoskonaleniem metody P.L. Ivanescu i współautorów (Rumunia), druga metoda jest nowa.

1. OKREŚLENIA

Niech x_s, $s = 1, 2,\ldots, n$, oznacza zmienną binarną, która może przyjmować tylko dwie wartości 0 i 1. Zmienna x_s może być również przedstawiona za pomocą negacji tej zmiennej zapisywanej jako $\bar{x}_s$. $\bar{x}_s = 0$ jeżeli $x_s = 1$ oraz $\bar{x}_s = 1$ jeżeli $x_s = 0$. Symbol $\tilde{x}_s$ oznacza x_s bądź $\bar{x}_s$.

Iloczyn (koniunkcja) niektórych lub wszystkich zmiennych $\tilde{x}_s$ będzie oznaczany przez X_i.

$$X_i = \tilde{x}_{i_1} \tilde{x}_{i_2} \ldots \tilde{x}_{i_p}, \quad i_m = 1, 2,\ldots, n; \quad m = 1, 2,\ldots, p;$$

$$p \leqslant n. \tag{1}$$

Ponadto $i_m \neq i_t$ dla $m \neq t$; $m, t = 1, 2,\ldots, p$. Indeks i jest numerem porządkowym koniunkcji, przy czym nie jest on związany z postacią koniunkcji, ale może być jej przyporządkowany w sposób dowolny.

Funkcja algebraiczna, której argumentami są zmienne x_s, $s = 1, 2,\ldots, n$, określona jest wzorem

$$F(x_1, \ldots, x_n) = \sum_{i=1}^{k} c_i X_i, \tag{2}$$

gdzie $c_i \neq 0$ są dowolnymi liczbami rzeczywistymi. Funkcja $F(x_1, \ldots, x_n)$ nazywana jest także funkcją pseudobulowską [1].

W artykule będą przedstawione dwie metody rozwiązywania równań i nierówności algebraicznych z niewiadomymi binarnymi o postaci

$$F(x_1, \ldots, x_n) > b, \tag{3}$$

$$F(x_1, \ldots, x_n) \geqslant b, \tag{4}$$

$$F(x_1, \ldots, x_n) = b, \tag{5}$$

gdzie b jest dowolną liczbą rzeczywistą.

2. METODA KOLEJNEGO ELIMINOWANIA KONIUNKCJI X_i

Metoda oparta jest na [1], gdzie podano algorytm rozwiązywania nierówności postaci

$$\sum_{i=1}^{k} c_i y_i \geqslant b. \tag{6}$$

Funkcja występująca po lewej stronie nierówności (6)

$$F(y_1, \ldots, y_k) = \sum_{i=1}^{k} c_i y_i \tag{7}$$

ma postać liniową względem swych argumentów y_i i jest zawsze przedstawiana w [1] w ten sposób, że $c_1 \geqslant c_2 \geqslant \ldots \geqslant c_k > 0$. Uzyskuje się to przez odpowiednie przekształcenie nierówności (6). Jeżeli

$$c = - \sum_{c_i < 0} c_i, \quad i = 1, 2, \ldots, k, \tag{8}$$

to nierówność (6) może być zapisana:

$$\sum_{c_i > 0} c_i y_i + \sum_{c_i < 0} (-c_i)(1 - y_i) \geqslant b + c$$

lub

$$\sum_{c_i > 0} c_i y_i + \sum_{c_i < 0} (-c_i) z_i \geqslant b + c, \quad i = 1, 2, \ldots, k. \qquad (9)$$

Rozwiązanie nierówności (9) metodą [1] polega na wyznaczeniu tak zwanej funkcji charakterystycznej, co jest równoważne zastąpieniu tej nierówności równaniem logicznym zawierającym po prawej stronie 1. Następnie dokonuje się podstawień

$$y_i = 1 - z_i = \bar{z}_i \quad \text{dla} \quad c_i < 0, \quad i = 1, 2, \ldots, k. \qquad (10)$$

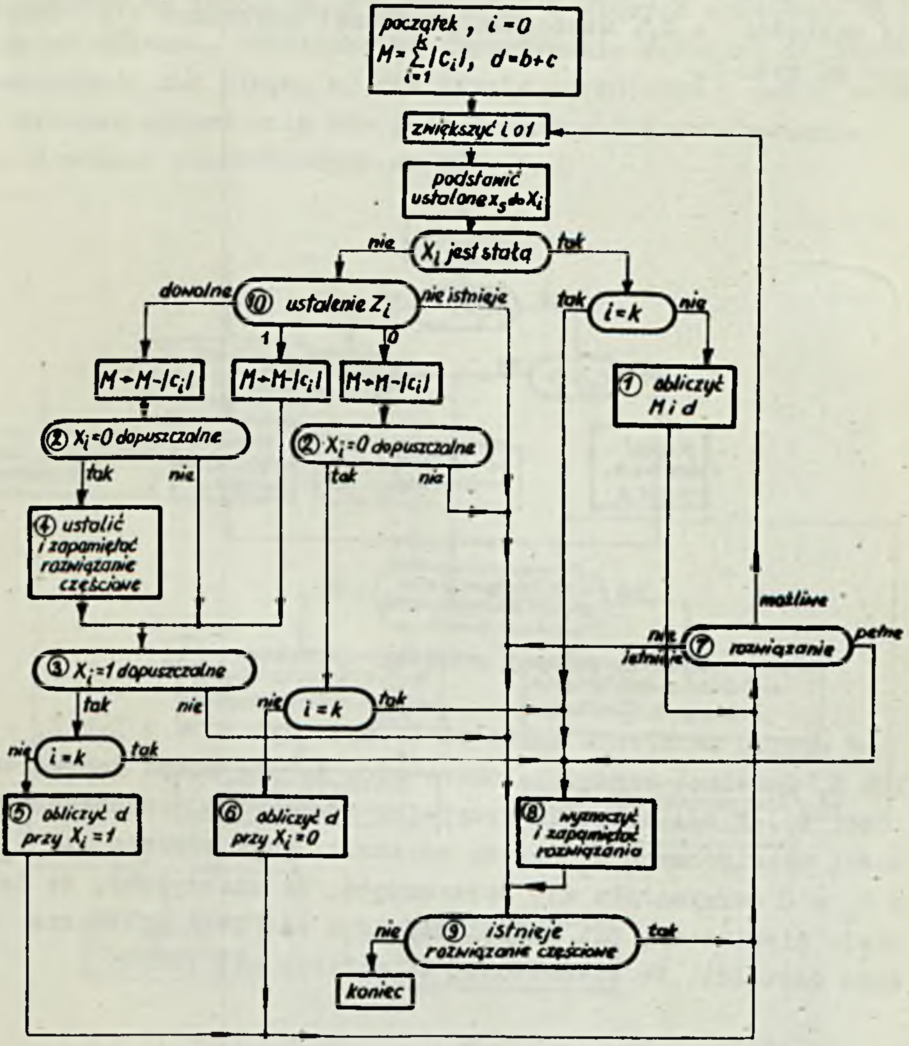
W przypadku rozwiązywania nierówności (4), gdzie F wyraża się wzorem (2), a X_i - wzorami (1), wartości niewiadomych $x_1, \ldots, x_n$ otrzymuje się z funkcji charakterystycznej po podstawieniu do niej zamiast y_i wzorów (1). Funkcja charakterystyczna przedstawia najpierw rozwiązania nierówności (4) dla przypadku, w którym X_i są traktowane jako zmienne niezależne. Powoduje to, że po dokonaniu podstawień (1) ilość rozwiązań może ulec zmniejszeniu. Ponadto mogą wystąpić powtarzające się rozwiązania, których eliminacja wymaga stosowania operacji logicznej pochłaniania. Niedogodności te eliminuje przedstawiona tu metoda, będąca udoskonaleniem metody [1]. Nie wymaga ona w ogóle tworzenia funkcji charakterystycznej i wprowadza w każdym kroku sprawdzanie czy wyznaczona wartość X_i jest dopuszczalna ze względu na niewiadome $x_1, \ldots, x_n$. Dzięki temu uzyskuje się bezpośrednio wartości tych niewiadomych.

W metodzie wykorzystuje się odpowiednio zmodyfikowany algorytm [1] rozwiązywania nierówności (4) podany dla funkcji $F(x_1, \ldots, x_n)$ o postaci liniowej z dodatkowym uwzględnieniem zadań (3) i (5). Dokonane modyfikacje umożliwiają zastoso-

wanie metody do obliczeń za pomocą maszyn cyfrowych, do czego nie jest przystosowany algorytm [1].

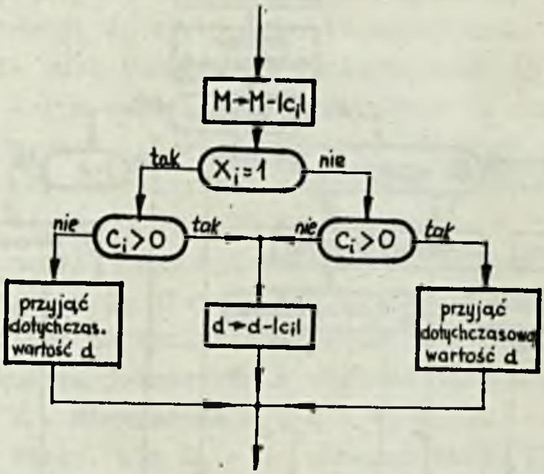
Załóżmy, że funkcja $F(x_1,\ldots, x_n)$ wyraża się wzorem (2), w którym $|c_1| \geqslant |c_2| \geqslant \ldots \geqslant |c_k| > 0$ oraz X_i, $i = 1,2,\ldots,k$, dane są wzorami (1). Zasadniczą cechą metody jest kolejne badanie koniunkcji X_i w porządku wzrastających indeksów i oraz eliminowanie tych koniunkcji z nierówności (3), (4) lub z równania (5). W tym celu każdej koniunkcji X_i przyporządkowuje się zmienną binarną Z_i. Wartość zmiennej Z_i jest określana tak, aby spełniona była rozwiązywana nierówność (równanie) i określa zarazem wartość X_i. Odpowiada to stosowaniu w [1] funkcji F w postaci liniowej (wprowadzenia za pomocą podstawień nowych zmiennych y_i i z_i), dlatego też ustalenie wartości Z_i oparte jest zasadniczo na wynikach uzyskanych w [1]. Otrzymana wartość Z_i oznacza jednocześnie wartość jaką może przyjąć sama koniunkcja X_i. Niewiadome x_s, $s = 1, 2,\ldots, n$, można wyznaczyć tylko wtedy, gdy $Z_i = 1$ (wówczas także $X_i = 1$) lub $Z_i = 0$, ale X_i zawiera tylko jedną zmienną $\tilde{x}_s$, która może być wtedy $\tilde{x}_s = 0$. W innych przypadkach (dla $Z_i = 0$) uzyskuje się tylko warunek $X_i = 0$, który należy zachować przy badaniu dalszych koniunkcji i wyznaczaniu pozostałych niewiadomych x_s. Warunek taki, który będzie nazywany dalej "ograniczeniem", otrzymuje się wówczas, gdy $Z_i = 0$ i odpowiadająca koniunkcja zawiera co najmniej dwie niewiadome x_s. Zmienne Z_i wprowadzono dla odróżnienia wartości jaką może przyjmować koniunkcja X_i od funkcji zmiennych x_s.

Ogólny schemat graficzny algorytmu przedstawia rys. 1. Jak widać z rysunku, badanie danej koniunkcji X_i rozpoczyna się od podstawienia do tej koniunkcji wartości zmiennych x_s, które zostały wyznaczone na podstawie wcześniej rozpatrzonych koniunkcji. Po tym podstawieniu może się okazać, że funkcja X_i nie zawiera żadnej zmiennej x_s, to znaczy jest równa stałej 0 lub 1. W takim przypadku nie ustala się już wartości zmiennej Z_i, ale określa warunki (operator ① dla $i < k$) zmieniające nierówność (równanie) po wyeliminowaniu danej koniunkcji X_i.



Rys. 1. Ogólny schemat algorytmu (metoda pierwsza)

Warunki te są związane przede wszystkim ze zmianą prawej strony nierówności (równania), po przeniesieniu na tę stronę stałej wartości $c_i X_i$. Szczegółowy schemat operatora (1) dany jest na rys. 2.



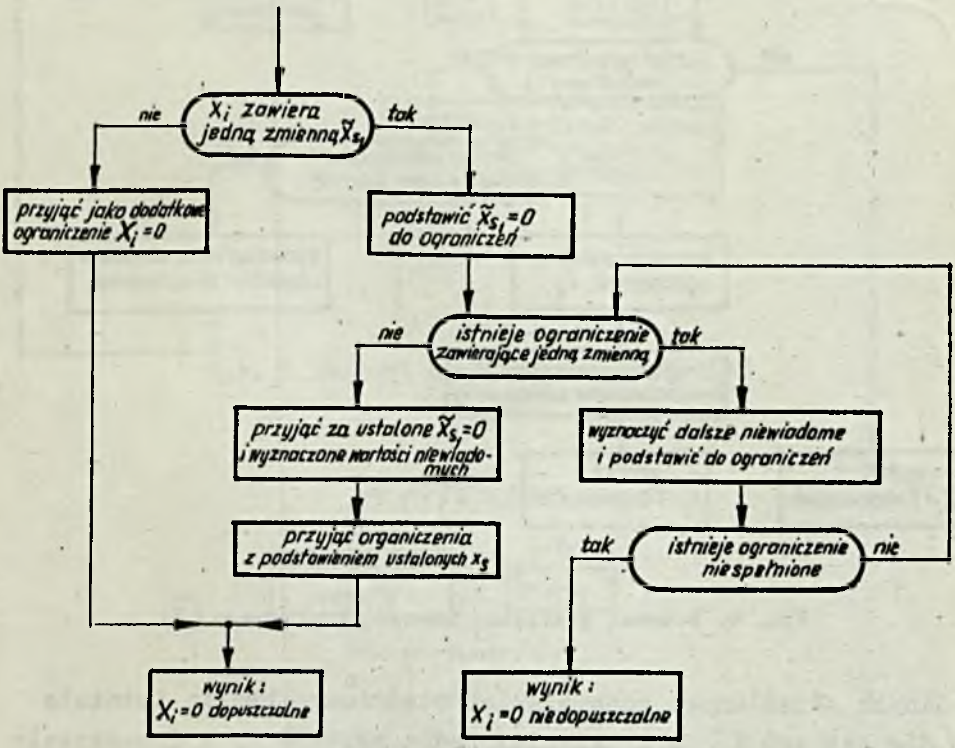
Rys. 2. Schemat graficzny operatora (1)

W wyniku ustalenia wartości Z_i może być $Z_i = 1$ lub $Z_i = 0$ lub Z_i dowolne, względnie okaże się, że nie można dobrać wartości Z_i. W tym ostatnim przypadku wyznaczone dotychczas wartości niewiadomych x_s należy odrzucić, gdyż wówczas dla $Z_i = 1$ i $Z_i = 0$ otrzymałoby się sprzeczność, co znaczyłoby, że istnieje niewiadoma, dla której uzyskuje się dwie sprzeczne ze sobą wartości. Po wyznaczeniu Z_i oblicza się liczbę

$$M = \sum_{j=i+1}^{k} |c_j| ,$$

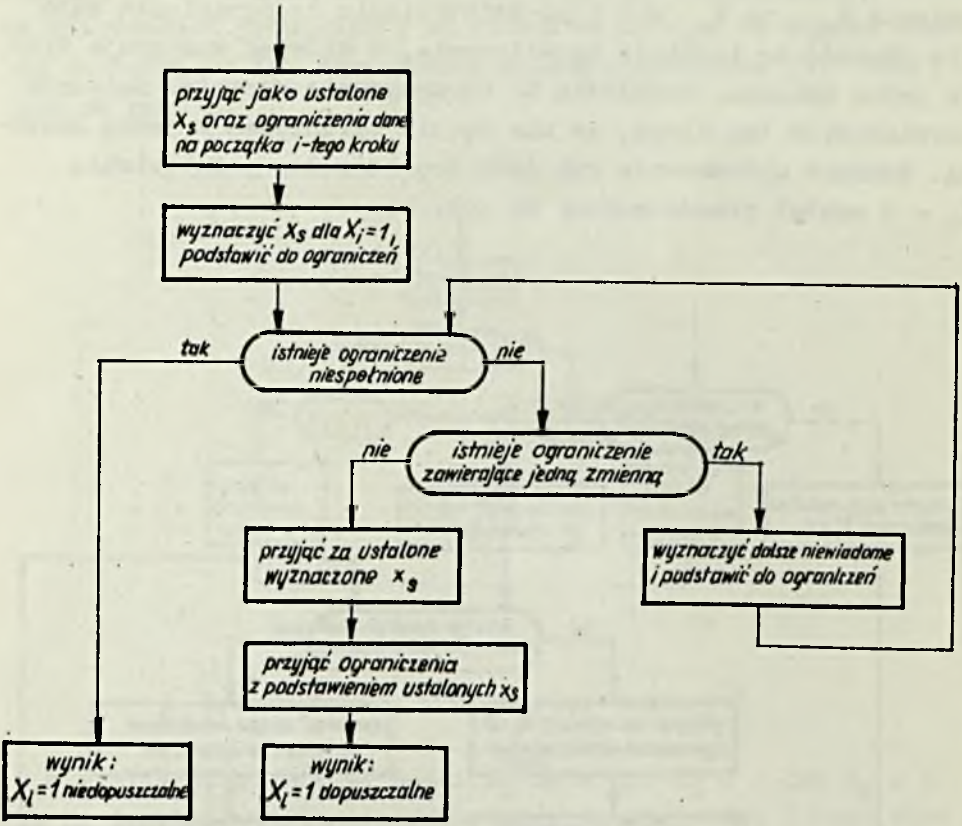
która oznacza maksymalną wartość funkcji F po wyeliminowaniu koniunkcji X_i. Z kolei należy sprawdzić, czy wartość X_i określona zmienną Z_i jest dopuszczalna ze względu na ograniczenia (warunki $X_j = 0$ dla $j < i$), otrzymane przy badaniu poprzednich koniunkcji. Schemat takiego sprawdzenia dla przypadku $Z_i = 0$

został podany na rysunku 3. Jeżeli X_i zawiera tylko jedną zmienną $\tilde{x}_{s_1}$, to $\tilde{x}_{s_1} = 0$ i po podstawieniu do ograniczeń może się okazać, że istnieją ograniczenia, w których występuje tylko jedna zmienna. Umożliwia to otrzymywanie wartości dalszych niewiadomych tak długo, aż nie będzie ograniczeń z jedną zmienną. Schemat sprawdzenia czy jest dopuszczalne podstawienie $X_i = 1$ został przedstawiony na rys. 4.



Rys. 3. Schemat graficzny warunku logicznego (2)

W przypadku, gdy Z_i może być dowolne i dopuszczalne jest $X_i = 0$, przewidziano zapamiętanie otrzymanych danych jako rozwiązania częściowego i przejście do wykonywania dalszych operacji. Rozwiązanie częściowe prowadzi do rozwiązania pełnego to jest takiego, które określa wartości wszystkich niewiadomych x_s, tylko wtedy, jeżeli przy kontynuowaniu obliczeń dla

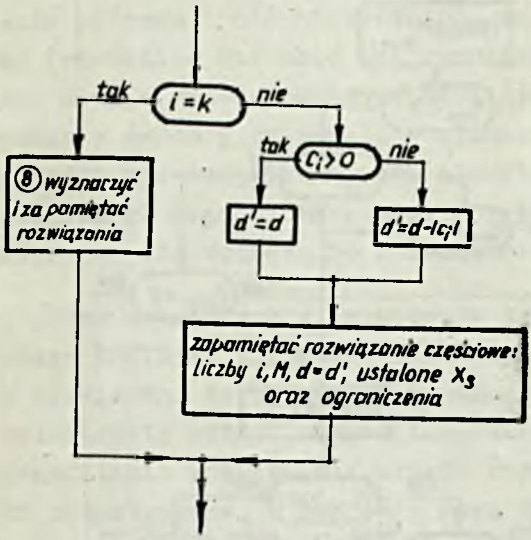


Rys. 4. Schemat graficzny warunku logicznego (3)

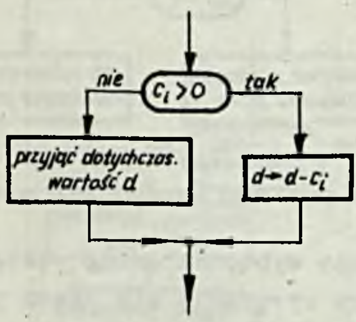
danych określonych rozwiązaniem częściowym będzie istniała (dla każdego i) przynajmniej jedna wartość Z_i i dopuszczalna wartość X_i. Część operatora (4) (rys. 5) związana z wyznaczeniem liczby d ma podobne znaczenie jak operator (1). To samo odnosi się do operatorów (5) (rys. 6) i (6) (rys. 7).

Może się zdarzyć, że przed zbadaniem wszystkich koniunkcji zostaną już wyznaczone wszystkie niewiadome x_s. Z tego względu, a także po to, aby uniknąć kroków, które nie doprowadzą do rozwiązania, został wprowadzony warunek logiczny (7) (rys. 8). Warunek ten nie jest niezbędny i może być pominięty

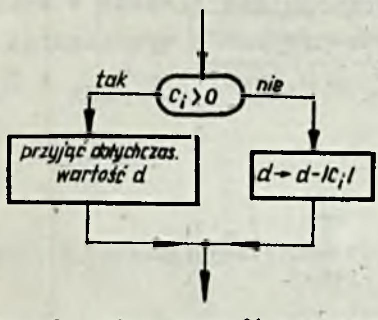
(należy wówczas przejść do następnego kroku - zwiększenia wartości i), jednak pozwala on na usprawnienie algorytmu.



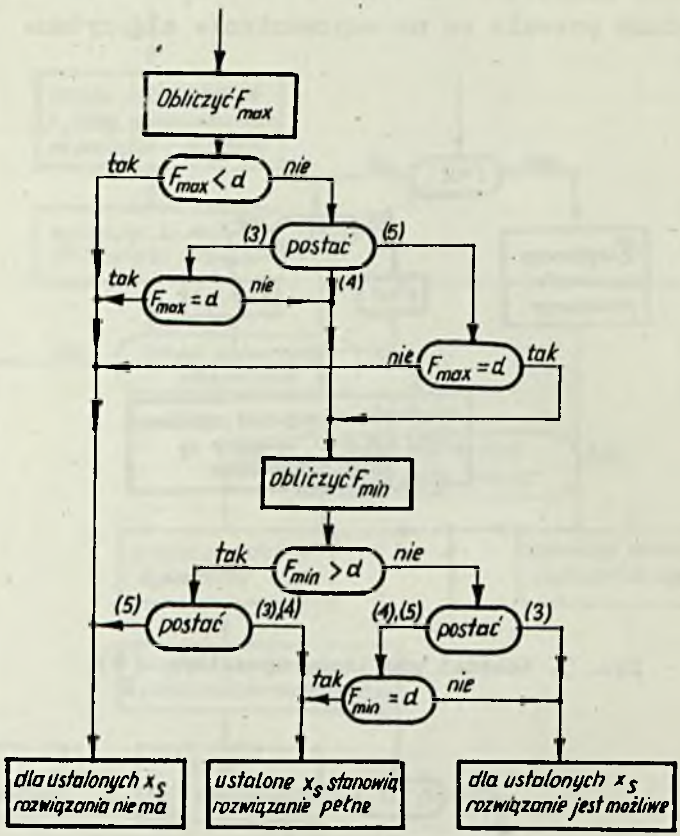
Rys. 5. Schemat graficzny operatora (4)



Rys. 6. Schemat graficzny operatora (5)



Rys. 7. Schemat graficzny operatora (6)



Rys. 8. Schemat graficzny warunku logicznego (7)

Warunek (7) uzależnia wybór dalszego postępowania od wartości F_{max} i F_{min}, które otrzymuje się przez podstawienie ustalonych już wartości niewiadomych x_s do pozostałych koniunkcji X_j $(j > i)$. Występujące jeszcze w koniunkcjach X_j niewiadome zakłada się w przypadku wyznaczania F_{max} – $\tilde{x}_s = 1$, a w przypadku F_{min} – $\tilde{x}_s = 0$, przy tym F_{max} i F_{min} ustala się ze wzoru:

$$F^l = \sum_{j=i+1}^{k} |c_j| X_j \qquad (11)$$

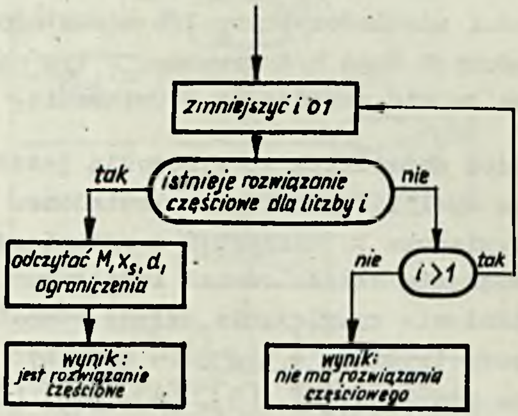
Warunek logiczny ⑦ uwzględnia wszystkie rozważane postacie nierówności (równania) - (3), (4), (5). Obliczenia są kontynuowane, jeżeli ustalone wartości x_s nie stanowią jeszcze rozwiązania pełnego i nie stwierdzono, że dla tych wartości nierówność (równanie) nie może być rozwiązana. Następuje wówczas dalszy krok - zwiększenie wartości i. Rozwiązanie pełne można uzyskać w dwóch przypadkach - wyznaczone zostały już wszystkie wartości niewiadomych x_s lub nieustalone jeszcze wartości niewiadomych mogą być dowolne. W tym ostatnim przypadku niewiadome te nie wchodzą do rozwiązania.

Niekiedy oprócz ustalonych x_s występują jeszcze ograniczenia, które muszą spełniać pozostałe niewiadome. W takim przypadku należy niewiadome te wyznaczyć z ograniczeń, co sprowadza się do rozwiązania układu równań logicznych (operator ⑧ - rys. 9). Zagadnienie rozwiązania układu równań logicznych nie będzie tu rozpatrywane, w tym celu mogą być stosowane dowolne metody, na przykład [2], [3], [4], [5], [6].



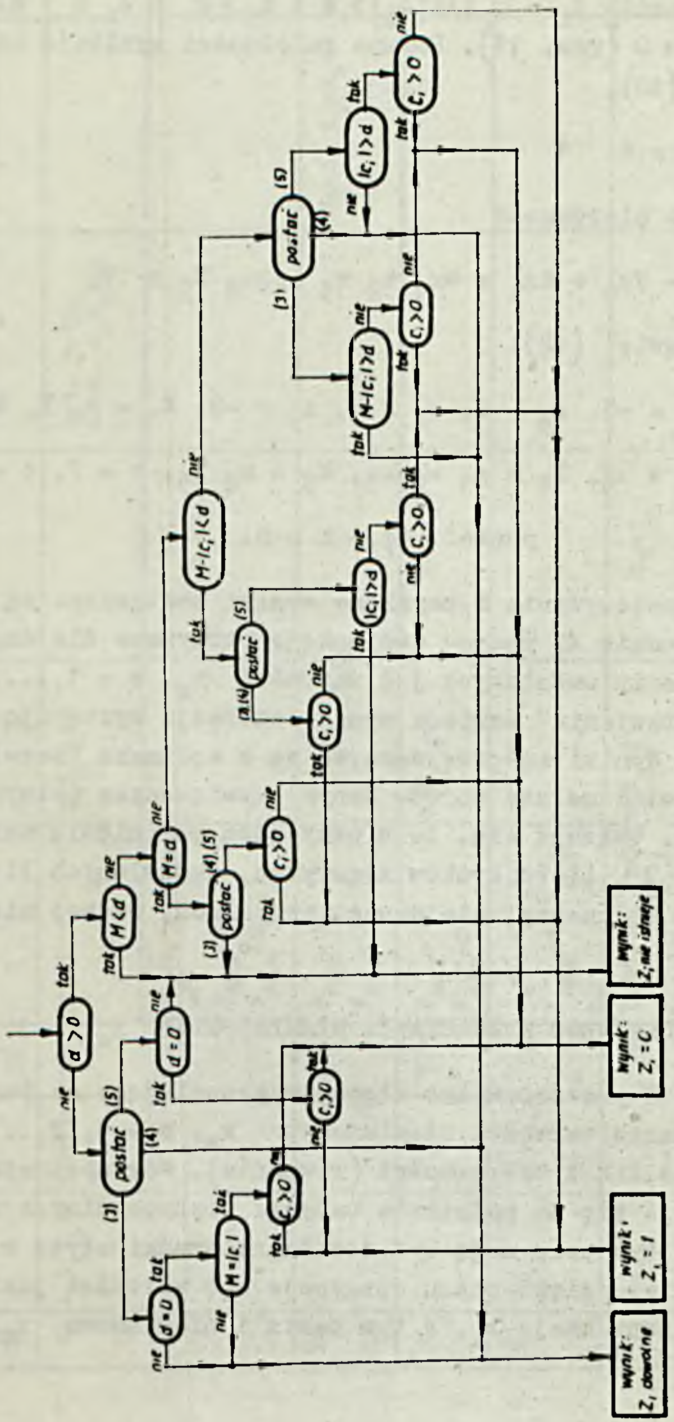
Rys. 9. Schemat graficzny operatora ⑧

Po otrzymaniu rozwiązania lub stwierdzeniu, że dla ustalonych wartości x_s rozwiązania nie ma, wybiera się rozwiązanie częściowe (jeżeli istnieje) i kontynuuje obliczenia (warunek logiczny (9) - rys. 10).



Rys. 10. Schemat graficzny warunku logicznego (9)

Na koniec zostanie omówiony schemat graficzny warunku logicznego (10) (rys. 11). Podobnie jak warunek logiczny (7) jest on uniwersalny i obejmuje rozwiązywanie zadań (3), (4), (5). Tylko część algorytmu (z rys. 1) dotycząca właśnie wyznaczenia Z_i oparta jest na pracy [1]. Podana w [1] metoda została tu przystosowana do obliczeń za pomocą maszyn cyfrowych i uzupełniona dla zadań (3) i (5). Na przykład w przypadku równania (5) dla $d < 0$ rozwiązań nie ma (rys. 11), co wynika ze wzoru (9), gdy uwzględni się analogie (2) i (7) i zastąpi znak nierówności znakiem równości. Jeżeli $d = 0$, to z uwagi na (10) powinno być $Z_i = 0$ dla $c_i > 0$ i $Z_i = 1$ dla $c_i < 0$. Dalej, jest rzeczą oczywistą, że gdy maksymalna wartość funkcji $M < d$, to nie ma rozwiązania, a gdy $M = d$, to $Z_i = 1$ dla $c_i > 0$ oraz $Z_i = 0$ dla $c_i < 0$ (rys. 11). Jeżeli $M > d$, ale $|c_i| > d$ i $M - |c_i| < d$, rozwiązania również nie ma. Rozwiązanie jest natomiast możliwe przy $|c_i| \leqslant d$ i wówczas $Z_i = 1$ dla $c_i > 0$, $Z_i = 0$ dla $c_i < 0$.



Rys. 11. Schemat graficzny warunku logicznego (10)

Wreszcie jeżeli $M > d$, $M - |c_i| \geqslant d$, Z_i jest dowolne dla $|c_i| \leqslant d$, ponadto $Z_i = 0$ dla $|c_i| > d$ i $c_i > 0$ i $Z_i = 1$ dla $|c_i| > d$ i $c_i < 0$ (rys. 11). Podane zależności wynikają ze wzorów (9) i (10).

P r z y k ł a d 1

Dana jest nierówność

$$13x_4 \bar{x}_5 \bar{x}_6 - 7x_6 + 6x_1 + 4x_1 x_2 x_3 - 2x_2 \bar{x}_5 \geqslant 7.$$

Na podstawie (12):

$$c_1 = 13,\ c_2 = -7,\ c_3 = 6,\ c_4 = 4,\ c_5 = -2,\ X_1 = x_4 \bar{x}_5 \bar{x}_6,$$

$$X_2 = x_6,\ X_3 = x_1,\ X_4 = x_1 x_2 x_3,\ X_5 = x_2 \bar{x}_5,\ b = 7,\ c = 9,$$

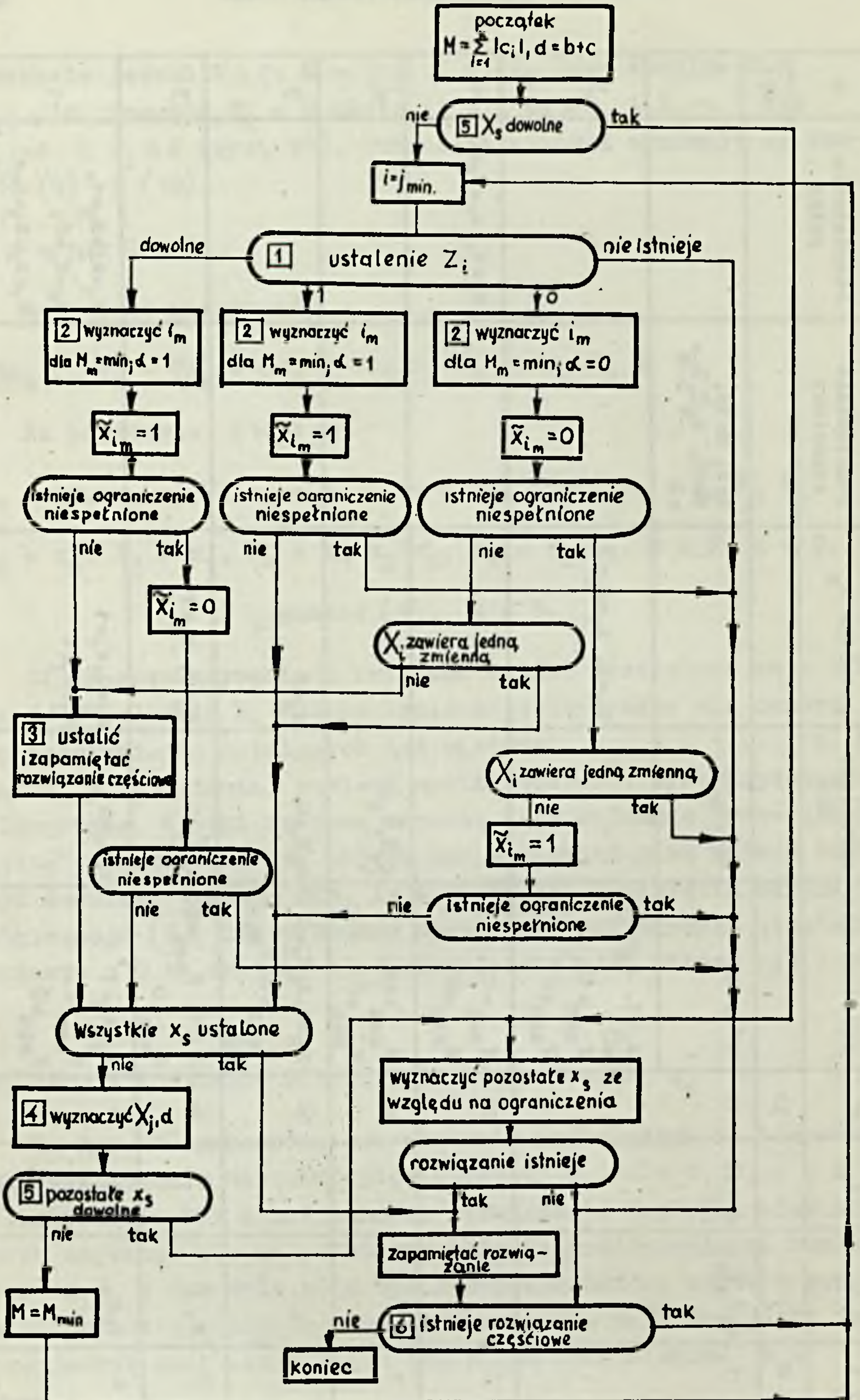
postać (4), $k = 5$.

Etapy rozwiązywania i uzyskane wyniki zestawione są w tabeli 1. W kolumnie X_i podano koniunkcje otrzymane dla danego i po podstawieniu ustalonych już wartości x_s, $s = 1, \ldots, 6$. Kolumna "podstawienia" zawiera wyniki operacji występujących w algorytmie. Wyniki końcowe zawarte są w kolumnie "rozwiązanie pełne". Niewiadome nie objęte danym rozwiązaniem pełnym mogą być dowolne. Okazuje się, że w przypadku pominięcia warunku logicznego (7), ilość kroków algorytmu, określonych liczbami i, wzrasta o 9 to znaczy, dla danego przykładu, więcej niż o 100%.

3. METODA KOLEJNEGO WYZNACZANIA NIEWIADOMYCH x_s

Na rys. 12 przedstawiono algorytm pozwalający na bezpośrednie wyznaczanie wartości niewiadomych x_s, $s = 1, 2, \ldots, n$ i eliminowanie ich z nierówności (równania). Wartości niewiadomych uzyskuje się na podstawie badania poszczególnych koniunkcji X_i. W tym celu mogą być stosowane środki użyte w poprzedniej metodzie, dzięki czemu otrzymuje się wartości jakie mogą przyjmować koniunkcje X_i, a tym samym i niewiadome x_s.

i	X_i	Z_i	M	podstawienia	ograniczenia	ustalone x_s	rozwiązanie częściowe	rozwiązanie pełne	d
0			32						16
1	$x_4\bar{x}_5\bar{x}_6$	dowolne	19	$d' = 16$	$x_4\bar{x}_5\bar{x}_6=0$		$i=1$, $M=19$, $d=16$, $x_4\bar{x}_5\bar{x}_6==0$		
				$x_4=1$, $x_5=x_6==0$ $F_{max}=12$, $F_{min}=0$	-	$x_4=1$, $x_5=x_6=0$			3
2	0		12	$F_{max}=12$, $F_{min}=0$	-			$x_4=1$, $x_5=x_6=0$	-4
1			19	$F_{max}=19$, $F_{min}=0$	$x_4\bar{x}_5\bar{x}_6=0$	-			16
2	x_6	0	12	$x_6=0, x_4\bar{x}_5=0$ $F_{max}=12, F_{min}==0$	$x_4\bar{x}_5=0$	$x_6=0$			9
3	x_1	1	6	$x_1=1$, $F_{max}=6$, $F_{min}=0$	$x_4\bar{x}_5=0$	$x_6=0$, $x_1=1$			3
4	x_2x_3	1	2	$x_2=x_3=1$ $F_{max}=2$, $F_{min}=0$	$x_4\bar{x}_5=0$	$x_6=0, x_1=1$ $x_2=x_3=1$		$x_1=x_2=x_3=1$, $x_4=x_6=0$; $x_1=x_2=x_3=x_5==1$, $x_6=0$	-1

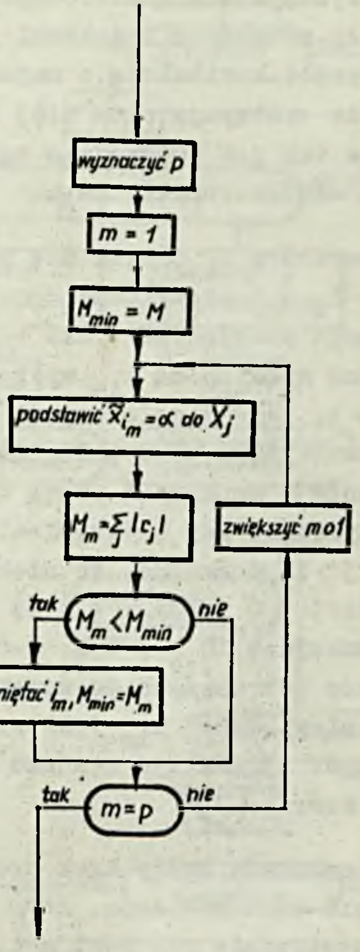


Rys. 12. Ogólny schemat algorytmu (metoda druga)

Koniunkcje, występujące z nierówności (równaniu) dla danego kroku, zostały oznaczone indeksami j. Badaniu podlega w pierwszej kolejności koniunkcja o najmniejszym indeksie j, ze względu na to, że występuje przy niej największy współczynnik $|c_j|$. Koniunkcje tak jak poprzednio są oznaczane indeksami według malejących współczynników $|c_j|$.

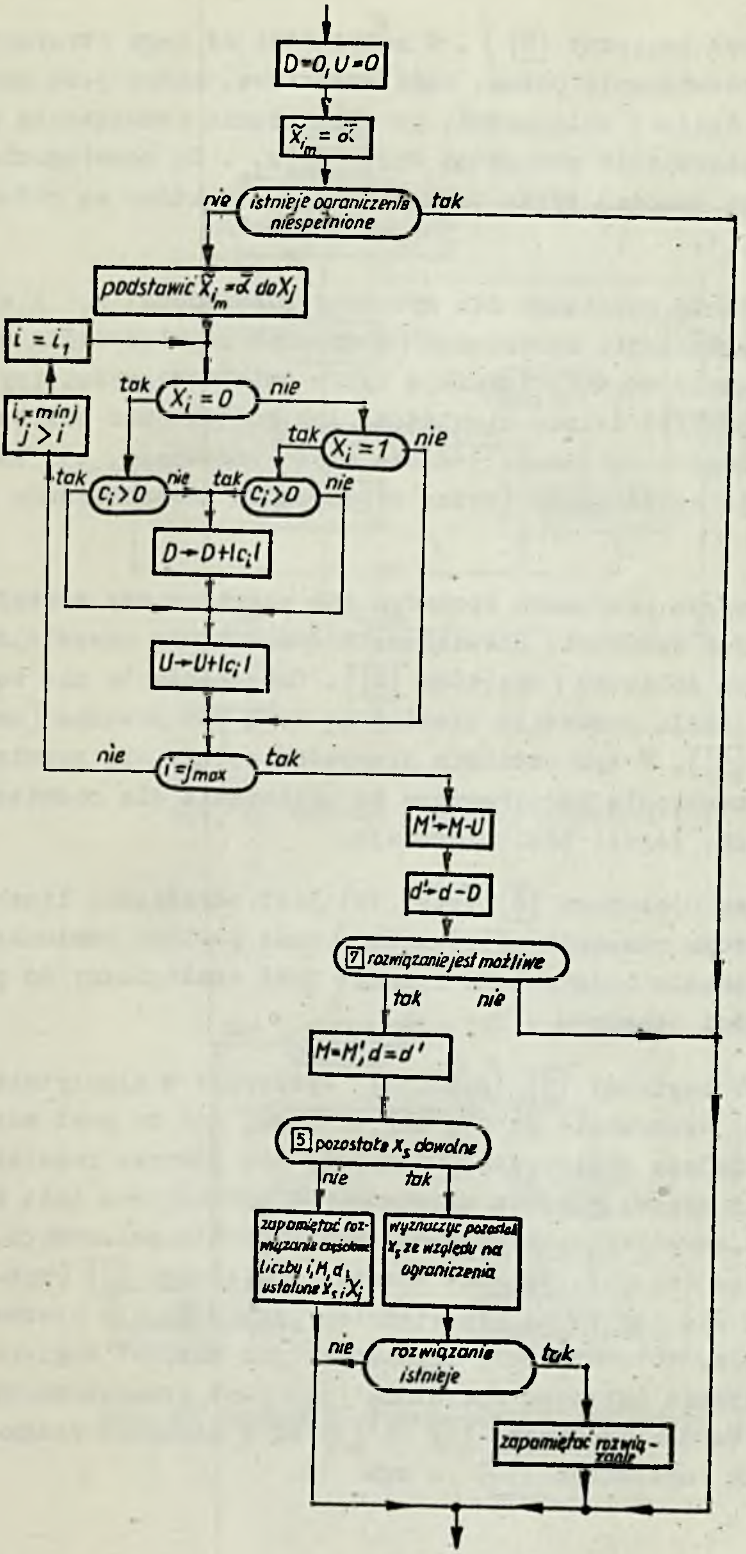
Określenie wartości Z_i odbywa się według algorytmu z rys.11. W przypadku gdy $Z_i = 1$ niewiadome występujące w koniunkcji X_i $(i = j_{min})$ mogą przyjmować tylko wartość 1 $(\tilde{x}_s = 1)$. Można by wybrać dowolną niewiadomą x_{im} występującą w koniunkcji X_i i przyjąć $\tilde{x}_{im} = 1$, jednak celowe jest znaleźć taką niewiadomą, która pozwoli zmniejszyć maksymalną wartość lewej strony nierówności (równania), oznaczaną przez M. Zmniejszenie liczby M następuje w przypadku, gdy podstawienie $\tilde{x}_{im} = 1$ do wszystkich koniunkcji X_j $(j > i)$ powoduje, że niektóre z koniunkcji przyjmują stałą wartość (0 lub 1); M jest sumą współczynników $|c_j|$ przy koniunkcjach $X_j \neq 0$ i $X_j \neq 1$. Dzięki temu zmniejsza się liczba kroków potrzebnych do uzyskania rozwiązania. Operator [2] wyboru niewiadomej x_{im} jest przedstawiony schematem graficznym na rys. 13. Na tym rysunku m i p oznaczają liczby występujące we wzorze (1).

W omawianej metodzie każdy krok prowadzi do rozwiązania i nie może wystąpić taka sytuacja, żeby po szeregu krokach okazało się, że rozwiązanie nie jest możliwe (chyba że rozwiązań nie ma w ogóle). Ponadto nie potrzeba wprowadzać ograniczeń w postaci równań logicznych stosowanych w metodzie z rozdziału 2. Natomiast uwidocznione na rys. 12 sprawdzanie ograniczeń odbywa się tylko w przypadku rozwiązywania układów nierówności - równań algebraicznych i logicznych. Rozwiązuje się wówczas tylko jedną nierówność (równanie), a pozostałe traktuje jako ograniczenie. W trakcie ustalania wartości niewiadomych algorytm z rys. 12 przewiduje sprawdzanie czy jest możliwe spełnienie ograniczeń. Szczegóły takiej metody rozwiązywania układu nierówności (równań) podane są w [7].



Rys. 13. Schemat graficzny operatora [2]

Jeżeli Z_1 jest dowolne lub $Z_1 = 0$ należy zbadać podstawienia $x_{i_m} = 1$, a także $x_{i_m} = 0$ (operator [3]). x_{i_m} jest natomiast jednoznaczne w przypadku, gdy $Z_1 = 0$ oraz gdy X_1 zawiera tylko jedną zmienną. Operator [3] ma na celu badanie drugiego rozwiązania dla wybranej niewiadomej x_{i_m}. Początkowe operacje dotyczą ustalenia liczb M' i d', a następnie przewidziane jest sprawdzenie, czy dla tych danych rozwiązanie jest możliwe (warunek logiczny [7]). Jeżeli tak, bada się jeszcze (rys. 14) czy niewyznaczone dotąd wartości x_s mogą być dowol-



Rys. 14. Schemat graficzny operatora [3]

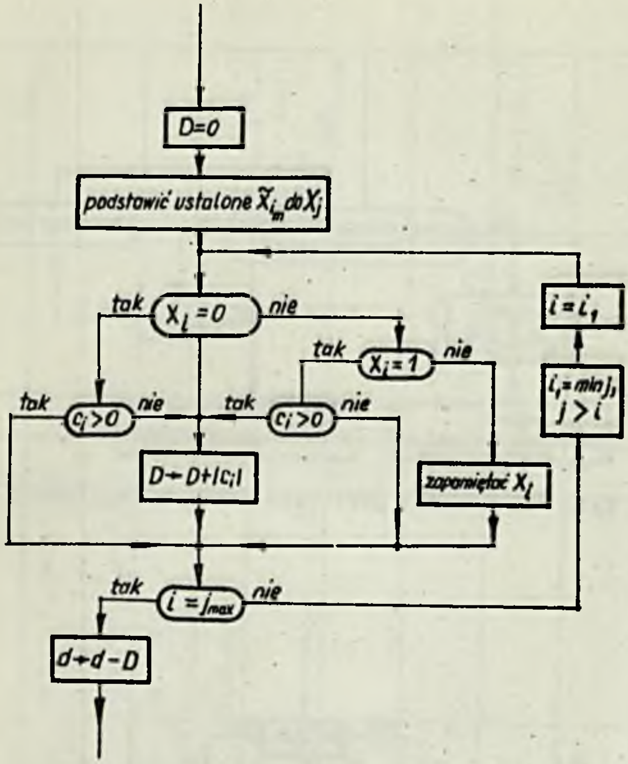
ne (warunek logiczny [5]). W zależności od tego otrzymuje się bądź rozwiązanie pełne, bądź częściowe, które jest kontynuowane w dalszej kolejności, po wyznaczeniu rozwiązania pełnego dla pierwotnie przyjętej wartości x_{i_m}. Do rozwiązania częściowego wchodzą tylko te koniunkcje X_j, które są różne od stałych 0, 1.

Po zbadaniu rozwiązań dla wybranej niewiadomej x_{i_m} i ewentualnym sprawdzeniu ograniczeń (przypadek rozwiązywania układu równań-nierówności) istnieją tylko dwie możliwości (rys. 12): należy wyznaczyć dalsze niewiadome lub kontynuować obliczenia dla dowolnego rozwiązania jeszcze niedokończonego, to znaczy rozwiązania częściowego (tylko w przypadku niespełnienia ograniczeń).

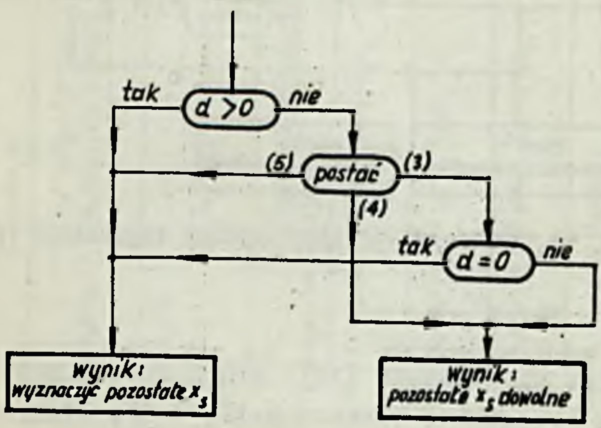
W pierwszym przypadku sprawdza się najpierw czy zostały wyznaczone już wszystkie niewiadome i ewentualnie określa dane do dalszych obliczeń (operator [4]). Obliczenia te nie są potrzebne jeżeli pozostałe niewiadome mogą być dowolne (warunek logiczny [5]). W tym ostatnim przypadku osiąga się rozwiązanie pełne, a następnie kontynuowane są obliczenia dla rozwiązań częściowych, jeżeli takie istnieją.

Zadaniem operatora [4] (rys. 15) jest określenie liczby d (prawa strona równania-nierówności) oraz postaci koniunkcji X_j po dokonaniu podstawień. Schemat jest analogiczny do początkowej części schematu z rys. 14.

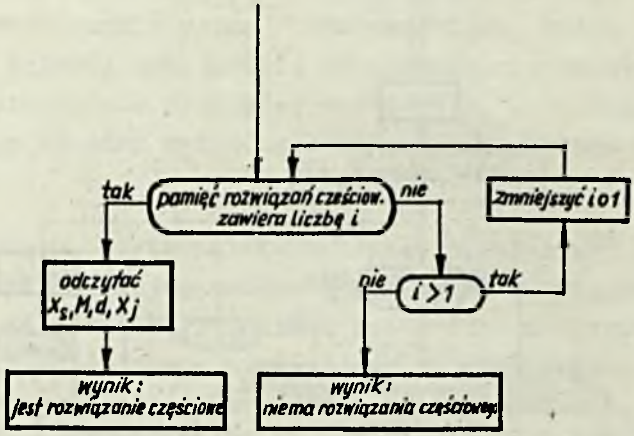
Warunek logiczny [5] (rys. 16) występuje w algorytmie (rys. 12) dwukrotnie po to, aby od razu, gdy to jest możliwe, przerwać dalsze obliczenia. Otrzymuje się wówczas rozwiązania pełne o mniejszej liczbie niewiadomych zawierające całą grupę rozwiązań określających wartości wszystkich niewiadomych, co jest bardzo dogodne. Schemat warunku logicznego [6] (rys. 17) nie różni się prawie od odpowiedniego schematu dla pierwszej metody (rys. 10). Wreszcie wspomniany już warunek logiczny [7] będący częścią składową operatora [3] jest przedstawiony na rys. 18. Warunki logiczne [7] i [5] są w zasadzie fragmentami warunku logicznego (10) z rys. 11.



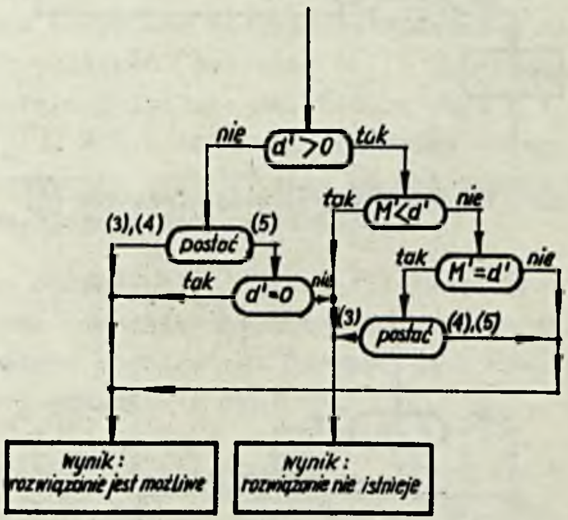
Rys. 15. Schemat graficzny operatora [4]



Rys. 16. Schemat graficzny warunku logicznego [5]



Rys. 17. Schemat graficzny warunku logicznego [6]



Rys. 18. Schemat graficzny warunku logicznego [7]

P r z y k ł a d 2

Rozwiązanie nierówności (12) drugą metodą przedstawia tabela 2. Dane początkowe wynoszą M=32, d=16. Otrzymane pełne rozwiązania są następujące:

Tabela 2

ustalone x_s	X_j	i	Z_i	$\tilde{x}_{1m}$	rozwiązanie częściowe	d	M	rozwiązanie pełne	uwagi
-	$X_1=x_4\bar{x}_5\bar{x}_6$, $X_2=x_6$, $X_3=x_1$, $X_4=x_1x_2x_3$, $X_5=x_2\bar{x}_5$	1	dowolne	$x_6=0$	-	9	25	-	
$x_6=0$,	$X_1=x_4\bar{x}_5$, $X_3=x_1$, $X_4=x_1x_2x_3$, $X_5=x_2\bar{x}_5$	1	dowolne	$x_4=1$	i=1, M=12, d=9, $x_6=x_4=0$, $X_3=x_1$, $X_4=x_1x_2x_3$, $X_5=x_2\bar{x}_5$	9	25	-	
$x_6=0$, $x_4=1$	$X_1=\bar{x}_5$, $X_3=x_1$, $X_4=x_1x_2x_3$, $X_5=x_2\bar{x}_5$	1	dowolne	$x_5=0$	i=1, M=10, d=7, $x_6=0$, $x_5=x_4=1$, $X_3=x_1$, $X_4=x_1x_2x_3$	-4		$x_6=x_5=0$, $x_4=1$	
$x_6=x_4=0$	$X_3=x_1$, $X_4=x_1x_2x_3$, $X_5=x_2\bar{x}_5$	3	1	$x_1=1$	-	3	6	-	z rozw. częśc. dla i=1, M=12, d=9
$x_6=x_4=0$, $x_1=1$	$X_4=x_2x_3$, $X_5=x_2\bar{x}_5$	4	1	$x_2=1$	-	3	6	-	
$x_6=x_4=0$, $x_2=x_1=1$	$X_4=x_3$, $X_5=\bar{x}_5$	4	1	$x_3=1$	-	-1		$x_6=x_4=0$, $x_3=x_2=x_1=1$	
$x_6=0$, $x_5=x_4=1$	$X_3=x_1$, $X_4=x_1x_2x_3$	3	1	$x_1=1$	-	1	4	-	z rozw. częśc. dla i=1, M=10, d=7
$x_6=0$, $x_5=x_4=x_1=1$	$X_4=x_2x_3$	4	1	$x_2=1$	-	1	4	-	
$x_6=0$, $x_5=x_4=x_2=x_1=1$	$X_4=x_3$	4	1	$x_3=1$	-			$x_1=x_2=x_3=x_4=x_5=1$, $x_6=0$	

1) $x_4 = 1, \quad x_5 = x_6 = 0,$

2) $x_1 = x_2 = x_3 = 1, \quad x_4 = x_6 = 0,$

3) $x_1 = x_2 = x_3 = x_4 = x_5 = 1, \quad x_6 = 0.$

Rozwiązania te pokrywają się z rozwiązaniami otrzymanymi pierwszą metodą, gdyż zapisane w tabeli 1 rozwiązanie $x_1 = x_2 = x_3 = x_5 = 1$, $x_4 = x_6 = 0$ zawiera się w rozwiązaniu 2.

4. WNIOSKI

1. Opracowano dwie oryginalne metody rozwiązywania nierówności (równań) algebraicznych z niewiadomymi binarnymi. Nie dotyczy to fragmentu ustalania możliwych wartości koniunkcji X_i (warunek logiczny (10) na schemacie z rys. 1 oraz [1] z rys. 12), opartego na [1]. Do określenia wartości X_i można stosować również odpowiednik warunku "ustalenie Z_i" podany w [7], który jest jednak mniej efektywny i może zwiększyć liczbę kroków potrzebnych do uzyskania rozwiązania.

2. W porównaniu z pracą [1], która jest bardzo cenna, omówione metody wprowadzają bezpośrednie wyznaczanie niewiadomych x_s $(s = 1, 2, \ldots, n)$, bez potrzeby stosowania zmiennych pomocniczych i "dwustopniowego" otrzymywania rozwiązania najpierw dla zmiennych pomocniczych, a następnie dla właściwych niewiadomych.

3. Nowością w podanych metodach jest wprowadzenie ograniczeń w postaci warunków $X_j = 0$ stanowiących układ równań logicznych, które rozwiązuje się przez sprawdzanie dla założonych wartości niewiadomych. Zastępuje to stosowanie funkcji charakterystycznych (metoda [1]) i dokonywane nad nimi operacje, niewygodne do obliczania za pomocą maszyny cyfrowej.

4. Sposób rozwiązywania układu nierówności-równań za pomocą przedstawionych metod sprowadza się do rozwiązywania tylko jednej nierówności (równania) i sprawdzania ograniczeń, którymi są pozostałe nierówności i równania układu, stanowi to zaletę w razie korzystania z maszyny cyfrowej. Poza tym algorytm mo-

że być stosowany bez żadnych zmian także wówczas, gdy rozwiązywany układ zawiera równania logiczne.

5. Metoda z rozdziału 3 jest efektywniejsza niż metoda podana w rozdziale 2, gdyż kolejne ustalanie wartości niewiadomych x_s zawsze prowadzi do rozwiązania, czego nie zapewnia metoda pierwsza (rozdz. 2). Pierwsza metoda może być stosowana zwłaszcza wówczas, gdy $k < n$ i odpowiednio k = liczba koniunkcji, n = liczba niewiadomych.

Literatura

[1] IVANESCU P.L., RUDEANU S.: Boolean Methods in Operations Research. First European Joint Conference of TIMS and ES. Warszawa 1966.

[2] BAZILEVSKIJ J.J.: Voprosy teorii vremiennych logiczeskich funkcij. Voprosy teorii matematiceskich masin. Moskwa 1958. sb. 1.

[3] GRIGORIAN J.G.: Algoritm reszenija sistemy logiczeskich uravnienij. Żurnał vycisletielnoj matematiki i matematiceskoj fiziki, 1962, t. 2, N 1.

[4] RUDEANU S.: O reszeni bulevych uravnienij. Sintez relejnych struktur. Trudy mieżdunaradnovo simpozjuma po teorii relejnych ustrojstv i koniecznych avtomatov /IFAC/, Moskwa 1965.

[5] PHISTER M.: Logical Design of Digital Computers. New York 1958.

[6] LIGMANOWSKI M.: Metoda rozwiązywania układu równań logicznych. Archiwum Automatyki i Telemechaniki, 1969, z. 1.

[7] LIGMANOWSKI M.: Układy równań i nierówności pseudologicznych. Prace Instytutu Łączności, 1970, nr 2 (w druku).

АЛГЕБРАИЧЕСКИЕ УРАВНЕНИЯ И НЕРАВЕНСТВА С БИНАРНЫМИ НЕИЗВЕСТНЫМИ

Резюме

Передается два метода решения алгебраических уравнений и неравенств, называемых тоже псевдобулевыми. Эти методы применимы к машинизации. Первый метод есть улучшением метода П.Л.Иванеску и соавторов (Румыния), второй метод является новым.

ALGEBRAIC EQUATIONS AND UNEQUALITIES WITH THE UNKNOWN BINARY VARIABLES

Summary

Two methods of solving of the algebraic equations and unequalities called also pseudoboolean equations and unequalities are given. They are adopted for the computer. The first one is a improvement of P.L. Ivanescu and coauthors' method and the second one is a original work of the author.

Pow. w IMM n. 450 pap. offset kl. III g. 80 zam. 31/71 GP-II-1416/68

Cena zł 60.—